Department of Physics and Astronomy, University of Canterbury,
Private Bag 4800, Christchurch, New Zealand

Optical Properties of the Lindquist–Wheeler Cosmology

Kane O'Donnell

MAPH480 Project, 2011

Supervisor: David Wiltshire

Abstract

We explore the properties of the Lindquist–Wheeler (LW) lattice cosmological model, and compare these to those of the standard Friedmann–Lemaître–Robertson–Walker (FLRW) one. Under certain assumptions, these models have similar large–scale dynamics. Clifton and Ferreira recently investigated the propagation of light in a spatially flat LW model, and found that certain optical properties differed. We find that their principal substantive result is flawed, and provide a correction. Notably, the redshift in the spatially flat LW model is the same as in the FLRW model. Therefore the spatially flat LW cosmology provides no alternative explanation for observations usually attributed to dark energy.

We consider light propagation in a LW model of negative spatial curvature, as it is suspected that such a model could explain these observations without the need to invoke dark energy. A theoretical discussion of such a hyperbolic LW universe is given, and a numerical prescription developed to investigate its optical properties. At the time of writing, these have not been fully implemented.

PACS: 98.80.-k, 98.80.Es

Contents

List of Figures

Chapter 1

Introduction

1.1 The inhomogeneous universe

One of the fundamental cornerstones of modern cosmology is the assumption that the universe is spatially homogeneous and isotropic. The apparent reason for the prevalence of this assumption is two–fold. Firstly, certain observational measurements, such as the isotropy of the cosmic microwave background (CMB), suggest that the universe was extremely homogeneous and isotropic when it first began. Secondly, the assumption of homogeneity and isotropy gives rise to considerable simplifications of the mathematical description of the universe. This leads to the highly successful Friedmann–Lemaître–Robertson–Walker (FLRW) model, which describes a universe that seems to conform extremely well with the one we observe at the present epoch.

This observational success has come at the expense of having to include nonbaryonic dark matter and dark energy as part of the standard cosmology. Most notably, in 1998 [1] the luminosity distances of supernovae provided definitive evidence that the expansion rate of the universe appears to have recently begun a phase of accelerated growth. This seemingly contradicts the universal attractiveness of gravity, and the versions of the FLRW model popular for most of the 20^{th} century. To remain viable, the standard cosmology was modified to include dark energy to drive this accelerated expansion. However, despite dark energy accounting for the majority of the energy–momentum in the universe, there are still no widely accepted physical explanations for its origin. This may be because dark energy is just an artefact of the overly simplistic assumptions involved in the standard cosmology — that the universe behaves as if it were homogeneous, in some average sense. Instead, it is possible that locally inhomogeneous models will have optical properties that result in the observed luminosity distances of supernovae and perceived accelerated expansion, without the need to invoke the poorly understood construct of dark energy.

There are further reasons for considering such inhomogeneous models. While the universe may have been spatially homogeneous and isotropic very early on, there is no a priori requirement that this hold true in the current epoch. Indeed, the present universe certainly does not appear locally homogeneous at all. Planets, stars, galaxies, voids, and filaments all form part of what appears to be a highly structured universe, and hence local homogeneity is clearly not a valid assumption. So, strictly speaking, the FLRW model (which does assume perfect local homogeneity), should be disregarded. The reason this is not immediately done follows from observations which suggest that on extremely large scales, of at least $110h^{-1}$ Mpc (where h is such that the Hubble constant $H_0 = 100h$ km/s/Mpc), the universe begins

to appear homogeneous, in some statistical sense. The dust approximation is then made — that gravity will act the same on these $110h^{-1}$ Mpc (and larger) "dust particles" in the same way it does on the perfect fluid that constitutes the mass–energy of the FLRW model.

This validity of this approximation is currently difficult to evaluate, given that it is not known how gravity acts on such scales. Of course, the current theory of gravity — general relativity — has passed all the tests it as been subjected to, with exceeding accuracy. But these tests are generally all within the solar system (or other isolated systems such as binary pulsars), and it remains to be confirmed whether general relativity is the correct description of gravity on large scales. Furthermore, even if general relativity is assumed to describe gravity on these scales, it is only ever applied in an average sense. This then raises the question as to whether the evolution of an averaged universe will be the same as the average of an exactly evolved universe.

The standard cosmology is currently facing an increasing challenge from those who are willing to query the aforementioned assumptions it involves. Notably, is the dust approximation valid, and how do the differing scales of (in)homogeneity impact on the FLRW model? At this stage, we conclude our brief overview of inhomogeneous cosmology, and refer the reader to [2, 3, 4, 5, 6, 7], and the references therein, for a general review of the past, present, and future of inhomogeneous cosmology, and the many ways in which such questions are being explored. We now focus on the specific approach of this paper — constructing a specific locally inhomogeneous lattice cosmology, and investigating its optical properties.

1.2 The Lindquist–Wheeler model

There have been many attempts at directly constructing locally inhomogeneous cosmologies that satisfy Einstein's general relativity, and that conform with observational data. The most well known of these are the Lemaître–Tolman–Bondi [8, 9] and Szekeres [10, 11] models, and consequent Swiss cheese models [12, 13]. However, these Swiss cheese models suffer from the fault that they are all based on an FLRW background. The purpose of this research was to investigate a specific model of the universe which was not based on such an exact FLRW background, and was explicitly locally inhomogeneous and discrete.

This model, proposed initially by Lindquist and Wheeler (LW) in 1957 [14], is a simple toy one that was motivated by the success of the Wigner–Seitz construction used to describe atomic solids [15, 16]. The LW model consists of a lattice of uniform masses (similar to a lattice of uniform atoms), composed by repeating a cell consisting of a central mass surrounded by empty space (similar to a Wigner–Seitz atomic unit cell, with a central atom surrounded by empty space). See Fig. 2.1a for an illustration of this. Furthermore, like Wigner and Seitz, Lindquist and Wheeler assumed spherical symmetry about each central mass, and therefore naturally chose the Schwarzschild geometry to describe each individual cell.

The analogy with an atomic lattice breaks down however, as the LW lattice is dynamical whereas an atomic lattice is not. That is, the cells of the LW universe may expand (since the derivative of the gravitational potential is nonzero at the cell boundaries), and hence the

entire lattice can also expand. This expansion can be quantified in terms of the radius a_{LW} of each individual spherical cell, and hence Lindquist and Wheeler took this to be the scale factor of their universe. Given a suitable choice of time coordinate, Lindquist and Wheeler then showed that their toy universe behaved similarly to the FLRW one, in the limit of many cells [14]. Thus the dynamical evolution of the LW model is similar to that of the FLRW one.[1]

One motivation for this research is that the optical properties of a cosmological model are arguably more important than the dynamical ones. All of our measurements depend directly on light rays, which in turn are directly affected by the medium through which they propagate. Notably, the problem of dark energy arises from observations of supernovae, and considerations of the luminosity distance–redshift relation. Some feel that there is in fact no such problem: while the supernovae themselves are behaving as they should dynamically, we are fooled into thinking they are not, by the distortion of their light rays by an inhomogeneous universe [17]. In a more general sense, one can can provide simple reasons as to why the optical properties of a LW universe should differ from those of an FLRW one. To do so, consider the Sachs optical equations [18], which describe how a bundle of light rays evolves through a given space–time. These are given by

$$\frac{\mathrm{d}\vartheta}{\mathrm{d}\lambda} + \vartheta^2 + \bar{\sigma}\sigma = -\frac{1}{2}R_{\mu\nu}k^\mu k^\nu \tag{1.1a}$$

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\lambda} + 2\sigma\vartheta = C_{\alpha\beta\gamma\delta}(\bar{t})^\alpha k^\beta (\bar{t})^\gamma k^\delta, \tag{1.1b}$$

where ϑ and σ are the expansion and shear scalars, respectively, and an over–bar denotes the complex conjugate. Furthermore, $R_{\mu\nu}$ is the Ricci tensor, $C_{\alpha\beta\gamma\delta}$ is the Weyl tensor, k^μ is a tangent vector to the null geodesic, λ is an affine parameter governing the propagation of the light ray, and t^μ is a unit null vector orthogonal to k^μ. In the FLRW universe, the assumption of a homogeneous perfect fluid energy density results in $R_{\mu\nu} \neq 0$ and $C_{\alpha\beta\gamma\delta} = 0$. In contrast, the LW universe is largely empty space, which has $R_{\mu\nu} = 0$ and $C_{\alpha\beta\gamma\delta} \neq 0$. A bundle of light rays will therefore evolve differently in the two models, as will the shear (which is notably absent in the FLRW model, as shown by (1.1b)). Furthermore, optical properties such as the angular diameter distance are directly related to the integrals of the shear and expansion scalars along the path of propagation. Therefore, these optical properties depend on the specific *local* properties of the universe ($\vartheta(\mathbf{x})$ and $\sigma(\mathbf{x})$ in this case), and not the average ones. Hence it seems quite reasonable that ignoring homogeneities, like is done in the FLRW cosmology, should give observational predictions that differ from those in a locally inhomogeneous universe.

These considerations apply to all inhomogeneous alternatives to the FLRW model. However, the LW model is simple enough to investigate directly. Indeed, Clifton and Ferreira (CF) recently carried out a detailed analysis of the optical properties of the spatially flat LW model [19]. Their results suggested that, in a spatially flat universe, the focusing of

[1] It is the same when the universe is spatially flat, but differs slightly when it is spatially curved, as discussed in §2.2.1.

light rays in the LW model is different to in the FLRW one, as expected from the previous arguments. Notably, they found that the relationship between the redshift and the distance measures in the LW model differed significantly from that of the FLRW models. As will be discussed in §4.1, it was found that some of these conclusions were incorrect, and that the luminosity distance–redshift relation is in fact the same as in the FLRW universe.

Nonetheless, the corrected LW model is still relevant, as it shows that a locally inhomogeneous universe can have different underlying optical properties, while still having dynamics which approach those of the current standard FLRW model. These optical differences do not remove the need for dark energy, however it may be the case that another inhomogeneous model will do so.

1.3 Outline of research

During initial investigations into the theoretical and numerical description of the LW model, as given by Clifton and Ferreira [19], I discovered that some of their conclusions were incorrect, as is discussed in §5. This was due to their theoretical prescription not being numerically implemented correctly, which lead to misleading and erroneous conclusions about the optical differences between the LW and FLRW cosmologies. This significant result will be published with Clifton and Ferreira.

The corrected CF model still suffered the same need for dark energy as the FLRW one, as discussed in §5. Yet the CF model was only applied numerically for a spatially flat universe. The corrections to the CF model, however, seemed to suggest that a hyperbolic universe could reduce the need for dark energy (see §2.2.1). Given that current observational data does not rule out the possibility of hyperbolic space [20], I hence explored the optical properties of a LW universe with negative spatial curvature. This involved generalising the (corrected) methods of Clifton and Ferreira to a hyperbolic LW universe. I believe I have done so successfully. Furthermore, I have developed a numerical program to model the optical properties of the LW universe in a hyperbolic space, similar to that of Clifton and Ferreira. However, at the time of writing, numerical results are yet to be obtained.

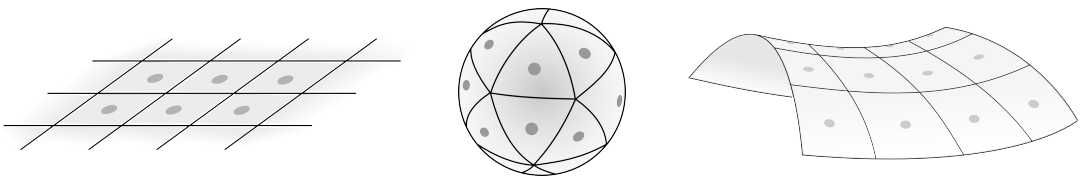
Chapter 2

The hyperbolic Lindquist–Wheeler universe

This chapter presents enough of the theory of the LW model [14] for the reader to gain a sufficient understanding of the conclusions we eventually make. The main focus is on the case of a universe with hypersurfaces of constant negative spatial curvature, and how modelling such universes will differ from the original approach given by Clifton and Ferreira [19]. For a more general introduction, the reader should refer to the original work of Clifton and Ferreira [19], and for further (mathematical) detail, the original work of Lindquist and Wheeler [14].

2.1 Lattices in a hyperbolic universe

The basic principle of the LW model is to approximate a FLRW universe of constant negative spatial curvature, by a discrete lattice of uniform cells. But how does one "tile" a universe of constant curvature with uniform cells?[1] In the case of an infinite universe with zero curvature, Euclidean geometry requires the space to be tiled with infinitely many cubic cells, in the obvious manner expressed in Fig. 2.1a. This was the method used by Clifton and Ferreira. However, when one considers nonzero spatial curvature the geometry becomes non–Euclidean, and the tilings become less obvious. In the case of a universe with positive spatial curvature, there are in fact only six possible complete tilings of the universe with regular cells. One can envisage the case of such a two–dimensional tiling as similar to a geodesic dome, as illustrated in Fig. 2.1b.



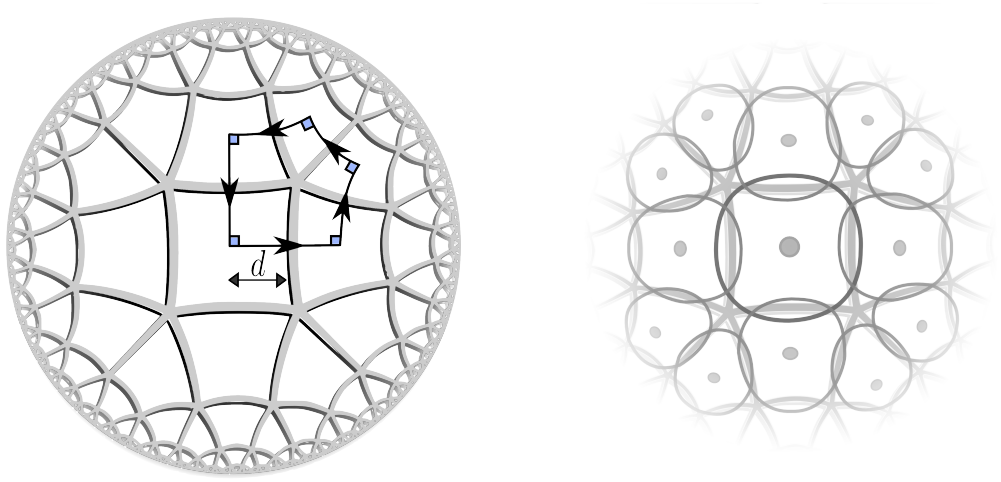
(a) **(b)** **(c)**

Figure 2.1 – *Examples of how to uniformly tile (a) flat, (b) positively curved, and (c) negatively curved two–dimensional spatial surfaces.*

[1] This report will only deal with universes with spatial hypersurfaces of constant curvature, so from now on all references to curvature shall be with this assumption implicit.

The case of a negatively spatially curved universe is similarly shown in Fig. 2.1c. In this two–dimensional hyperbolic geometry, one can represent all spaces of constant negative curvature by the Poincaré disk, as in Fig. 2.2a. This figure also illustrates a simple two–dimensional tiling of the Poincaré disk, using hypersquares. One can now appreciate why there is "more space" in hyperbolic space — in this case, five hypersquares are squeezed around each vertex, as opposed to the usual four in Euclidean geometry.

In two dimensions, a result from hyperbolic geometry shows that the Poincaré disk can be tiled by infinitely many different patterns [21] of regular tiles. However, In three–dimensional hyperbolic space there are only four possible complete tilings of the hyperbolic universe [21]. In this report, we choose to investigate the simplest hypercubic tiling for reasons of simplicity and adaptability to the method of Clifton and Ferreira. This is because it is similar to the flat case, except with the caveat that five cubes now meet at each edge, instead of four in flat space. This is a little more difficult to illustrate, however, Fig. 2.2a can be taken as a two–dimensional slice of the larger three–space.



(a) **(b)**

Figure 2.2 – *The Poincaré disk representation of a hyperbolic two–dimensional universe. A tiling is shown in (a), with hypersquares of dimension d. The path drawn illustrates how if one travels a distance* $2d$ *from the origin, turns left by* $90°$*, then repeats, it will take five steps to return to the origin. The Schwarzschild cell approximations to each cell are shown in (b), with the dots being the central masses. As discussed in the text, the boundaries are not circular in this representation.*

The property that there are five hypersquares around each vertex in Fig. 2.2a constrains the dimensions of the tiles to be used. Given that the map to the Poincaré disk (or Poincaré ball in three dimensions) is conformal, the interior angles of each hypersquare must be $72°$. As shown in Appendix A.2, this uniquely specifies the distance d illustrated in Fig. 2.2a to be $d = \sqrt{\sqrt{5}+2} - \sqrt{\sqrt{5}+1}$.

Does this unique d in the Poincaré ball have a corresponding unique distance D in the real universe? As one of the fathers of hyperbolic geometry, Carl Gauss, once stated

> *"I have satisfactorily constructed this* [hyperbolic] *geometry for myself so that I can solve every problem, except for the determination of one constant, which cannot be ascertained a priori... If non–Euclidean geometry were the true geometry, and if this constant were comparable to distances which we can measure on Earth or in the heavens, then it could be determined a posteriori. Hence I have sometimes in jest expressed the wish that euclidean geometry is not true. For then we would have an absolute a priori unit of measurement."* [22].

To explain Gauss's meaning, consider the scale of a hyperbolic universe. The *"constant"* Gauss speaks of is the radius of curvature — essentially how far one must go before the effects of curvature are noticed. Certainly, this radius of curvature is not *"comparable to distances which we can measure on Earth or in the heavens"* as, for example, we measure the angles within a triangle to be 180° on such scales. Hence this radius of curvature cannot be determined a posteriori, and is another degree of freedom in our cosmological models. Another way to think of this is as a scaling problem — one could define the metre as being the distance light travels in $1/c$ seconds, or as the distance light travels in $1/(2c)$ seconds, and there would be no physically observable consequence, as all other units would change accordingly. Therefore, whatever metric is used to describe the universe, there should be some initial freedom to set a scale — to define 1m as the distance light travels from some t_a to some t_b. Of course, once this choice is made, then Gauss is correct in that there is now a fixed *"unit of measurement"*.

An obvious way to fix this length scale is to choose some value for $a_0 = a(t_0)$ for the scale factor at the present epoch. (Given the spatial Gaussian curvature is $K = -1/a^2$, this is equivalent to specifying the spatial curvature of the universe at the present epoch.) Once this is fixed however, the constraints imposed by the hyperbolic tiling must also be considered — notably the dimension of the hypercube (in Poincaré coordinates) is

$$D = \ell_{r,p}(d) = 2a(t)\tanh^{-1}(d)\,, \tag{2.1}$$

where $d = \sqrt{\sqrt{5}+2} - \sqrt{\sqrt{5}+1}$, as discussed in §A.1. This dimension now makes clear the *"a priori unit of measurement"*, in the following manner. If one starts at the origin and travels a distance $2D$ in the x-direction, and then turns left by 90°, and travels another $2D$ and turns left again, and so on, it will take *five* steps to return to the origin, as illustrated in Fig. 2.2a. This is the *unique* distance D for which this will work in the hyperbolic universe.

2.1.1 Determining the cell sizes

Consider a hyperbolic universe specified by the Robertson–Walker metric

$$\mathrm{d}s^2 = -c^2\mathrm{d}t^2 + a(t)^2\left(\frac{\mathrm{d}r^2}{1+r^2} + r^2\mathrm{d}\Omega^2\right), \tag{2.2}$$

where $a(t)$ is the scale factor, as discussed in Appendix A.1. The Friedmann equation is just the 00 component of the Einstein field equation

$$R_{00} - \frac{1}{2}g_{00}R = \frac{8\pi G T_{00}}{c^4}, \tag{2.3}$$

where R_{00}, g_{00}, and T_{00} are the 00 components of the Ricci tensor, the metric tensor, and the energy–momentum tensor respectively, R is the Ricci scalar, and G is the gravitational constant. For the metric (2.2),

$$g_{00} = -c^2\,, \qquad R_{00} = -\frac{3\ddot{a}}{a}\,, \qquad R = \frac{6\left(\ddot{a}a + \dot{a}^2 - c^2\right)}{a^2c^2}\,. \tag{2.4}$$

Also, given $T_{\hat{0}\hat{0}} = \rho c^2$ in an orthonormal frame, then $T_{00} = \Lambda^{\hat{\alpha}}{}_0\Lambda^{\hat{\beta}}{}_0 T_{\hat{\alpha}\hat{\beta}} = \rho c^4$, (where hats indices denote an orthonormal frame, and $\Lambda^{\hat{\alpha}}{}_0 = \mathrm{diag}(c,1,1,1)$). Combining these results, the 00 component of the Einstein field equation becomes the usual Friedmann equation for a hyperbolic universe

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{8\pi G\rho}{3} + \frac{c^2}{a^2}\,, \tag{2.5}$$

or in the alternate notation, for a matter–dominated universe,

$$1 = \Omega_m + \Omega_k \quad \text{where} \quad \Omega_m = \frac{8\pi G\rho}{3H^2}\,, \quad \Omega_k = \frac{c^2}{a^2H^2}\,, \tag{2.6}$$

and $H = \dot{a}/a$ is the Hubble constant. Letting m be the mass of each hypercubic cell (which is constant as it is assumed that no matter flows between cells), then it follows that

$$V_0 = \frac{8\pi G m}{3\Omega_{m0}}\,, \tag{2.7}$$

where V_0 is the volume of each cell, Ω_{m0} is the value of the matter density parameter, and H_0 is the value of the Hubble constant, all at the present time. However, the volume V_0 of the hypercube can also be calculated directly from the metric via

$$V_0 = 8\,a_0^3\,V_p\,, \tag{2.8}$$

where $V_p \approx 0.289$ is the volume of the hypercube in the Poincaré ball, as discussed in Appendix A.3. Equating the expressions (2.7) and (2.8) for the volume V_0 determines the scale factor a_0 at the present day to be

$$a_0 = \left(\frac{\pi G m}{3V_p\Omega_{m0}H_0^2}\right)^{\frac{1}{3}}. \tag{2.9}$$

To summarise, for a given mass m in each cell, and a curvature specified by Ω_{m0}, this a_0 is the scale factor that determines the absolute scale of the "real" universe. It is determined by the requirement that the hyperbolic universe be tiled completely (that is, that five hypercubes exist around the edge of each hypercube), and that the density of the lattice universe matches that of the current universe, such that the Friedmann equation (2.5) is satisfied *on average*. The important consequence for the "real" hyperbolic universe is that it determines the *"a priori unit of measurement"* of Gauss, and hence the dimension D_0 of the hypercubes needed to tile the universe at the present epoch:

$$D_0 = 2a_0 \tanh^{-1}(d)\,, \tag{2.10}$$

where $d = \sqrt{\sqrt{5}+2} - \sqrt{\sqrt{5}+1}$.

2.1.2 A coordinate description

In a spatially flat Euclidean space, the coordinate axes, and straight lines parallel to them, are geodesics. However, in hyperbolic space, geodesics are curved with respect to the flat space coordinates of any chart. For example, in the Poincaré disk, geodesics correspond to sections of arcs that intersect the boundary orthogonally (the boundaries of the hypersquares in Fig. 2.2 are such geodesics). For the space–time to have the same symmetry properties as that of flat space, one must work in coordinates X, Y, and Z in the hyperbolic space, such that these axes, and curves "parallel" to them, are also geodesics. Such hyperbolic geodesics coordinates are discussed and derived in Appendix A.1.

These coordinates are further useful, as they determine the boundary of the hypercube in the same manner as in Euclidean space — that is, $X^i = \pm D_0$. While the metric is indeed different, as given in Appendix A.1, the distance along a coordinate axis is the same in both the (x, y, z) and (X, Y, Z) coordinates, so the D_0 of (2.10) still determines the size of the cell. However, general distances will differ in the (X, Y, Z) coordinates, so if one wished to keep track of, for example, the absolute position in the hyperbolic universe of the current cell, one would have to use the (X, Y, Z) metric. As discussed in Appendix A.1, this is considerably more complicated, and seems to have no simple solution for the proper distance. Thankfully, we do not require the calculation of such distances in our numerical procedure, and these (X, Y, Z) coordinates serve only the role of determining the spherical polar coordinates (r, θ, ϕ) that are used:

$$X = r\sin\theta\cos\phi\,, \quad Y = r\sin\theta\sin\phi\,, \quad Z = r\cos\theta\,, \quad r^2 = X^2 + Y^2 + Z^2\,. \tag{2.11}$$

This also explains why the boundaries of the Schwarzschild cells in Fig. 2.2b do not appear circular — the underlying coordinate system in that figure is the Euclidean (x, y, z), and not the (X, Y, Z) on which the Schwarzschild geometry is placed.

2.2 The Schwarzschild approximation

Now that the hypercubic tiling of the universe is specified, the analogy to the Wigner–Seitz construction is extended, and spherical symmetry is assumed, as is illustrated in Fig. 2.2b. The boundary of each hypercubic cell is replaced with a spherical one, and the space–time geometry within each cell is approximated to be Schwarzschild, with the central mass m being the mass of the original hypercubic cell (though the mass m is now a point mass, i.e., a black hole). The validity of such approximations are considered in §5.

There are two natural choices for the size of the spherical boundary. Firstly, the spherical cells could be chosen so that they are only just touching each other, with no overlap. Alternatively, the size of the spherical cell could be specified such that the volume remains the same as that of the original cell. As discussed in [14] and [19], the second method is the most suitable. This is because the density will now be the same as in the original cells, and there will be an averaging out of the opposing effects in the overlapping and uncovered regions of Fig. 2.2b, as discussed in §3.2.1. The original numerical implementation of Clifton and Ferreira actually involved something similar to the first method, and this was the source of the difference in optical properties between the LW and FLRW models. This is discussed more in §4.1.

Within the spherical boundary of an individual cell, the Schwarzschild geometry is used to approximate the true geometry. The Schwarzschild geometry is specified by the metric

$$\mathrm{d}s^2 = -c^2\Delta\mathrm{d}t^2 + \Delta^{-1}\mathrm{d}r^2 + r^2\mathrm{d}\Omega^2\,, \tag{2.12}$$

where $\Delta \equiv 1 - 2Gm/(c^2 r)$ with m the mass in the original hypercubic tile, and $\mathrm{d}\Omega^2 = \mathrm{d}\theta^2 + \sin^2\theta\mathrm{d}\phi^2$. However, considering Fig. 2.3a, it becomes apparent that this geometry cannot be used in each cell, as it creates matching problems for the overall geometry. Specifically, the lines of constant Schwarzschild time t from different cells are not tangential to each other at the boundaries, as in Fig. 2.3a, and hence t cannot be used as a cosmological time coordinate to describe the global space–time.

Lindquist and Wheeler resolve this problem by transforming to a new time coordinate that is tangential at the intersection of the cell boundaries. Clifton and Ferreira generalised this by finding a new time coordinate τ that overlaps completely throughout adjacent cells, and hence over the entire lattice universe, as illustrated in Fig. 2.3b. This then gave a sensible notion of a cosmological time, which was then applied to spaces of arbitrary constant spatial curvature.

Specifically, the transformation to the new time coordinate τ is given by

$$c\,\mathrm{d}\tau = c\sqrt{E}\,\mathrm{d}t - \frac{\sqrt{E-\Delta}}{\Delta}\,\mathrm{d}r\,, \tag{2.13}$$

where E is a positive constant to be defined later. The metric of each spherical cell then becomes

$$\mathrm{d}s^2 = -\frac{\Delta c^2}{E}\mathrm{d}\tau^2 - \frac{2c}{E}\sqrt{E-\Delta}\,\mathrm{d}\tau\,\mathrm{d}r + \frac{\mathrm{d}r^2}{E} + r^2\mathrm{d}\Omega^2. \tag{2.14}$$

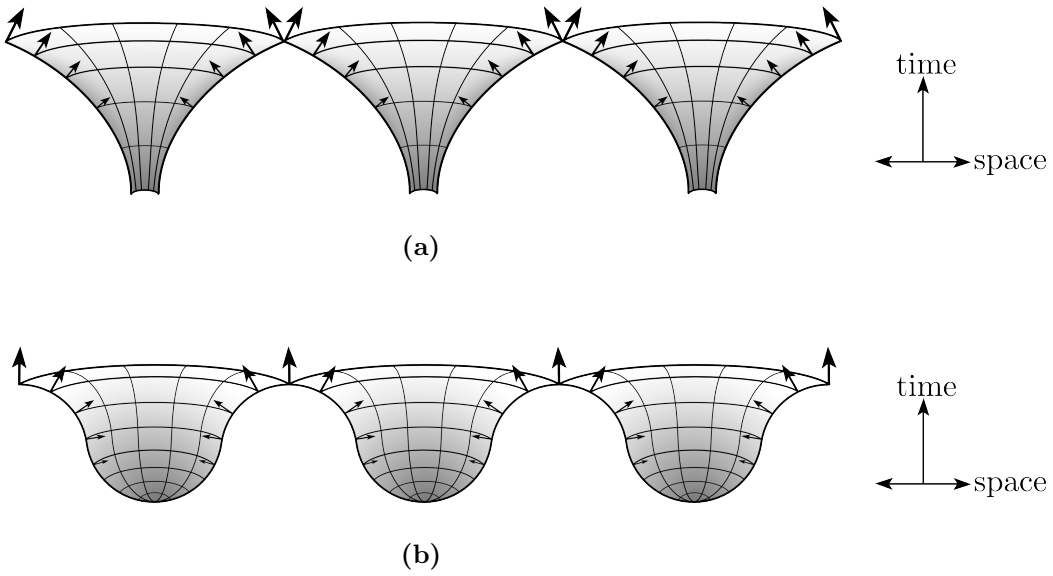


Figure 2.3 – *Tiling a universe with standard Schwarzschild cells as in (a) is unsuitable, as the surfaces of constant t are not tangential at the intersection of cells. Changing to τ as in* (2.13) *leads to surfaces of constant τ being tangential throughout the lattice, as in (b), and hence τ is a suitable cosmological time.*

For the spatially flat case, $E = 1$, and this then becomes the Gullstrand–Painlevé coordinate system [23, 24], which Clifton and Ferreira eventually use. It follows from (2.14) and the conserved quantity of the time–like killing vector, that any radially out–falling time–like particle must satisfy

$$\left(\frac{\mathrm{d}r}{\mathrm{d}\tau}\right)^2 = c^2(E - \Delta). \tag{2.15}$$

In this context, it can be shown that the constant $E = \mathcal{E}^2/c^4$, where $\mathcal{E}$ is the energy per unit mass of the particle at spatial infinity.

Since the boundaries of the LW cells have been taken to be spherical, and may be arbitrarily expanding or contracting, then any particle on such a dynamic boundary will have a four-velocity

$$u^\mu = \left(1, c\sqrt{E - \Delta}, 0, 0\right), \tag{2.16}$$

which clearly satisfies (2.15). Any vector in a surface of constant τ must be of the form $n^\mu = c\,(0, n^r, n^\theta, n^\phi)$, and it is easily checked that $u^\mu n_\mu = 0$. Hence any surface of constant τ is orthogonal to to the spherical Schwarzschild cell boundary, which is expanding with a velocity given by (2.16). This holds for a Schwarzschild boundary of any radius, so surfaces of constant τ are always orthogonal to the boundaries of the Schwarzschild cells. Therefore, when the spherical Schwarzschild boundaries of two cells are touching (at the boundary between the overlap region and the empty region), the surfaces of constant τ between the neighbouring cells will be tangential, as illustrated in Fig. 2.3b. However, within the overlap region or the uncovered region, this will not be true exactly, but only approximately so [19].

As discussed in §3.2.1, the effect in the uncovered region is opposite that in the overlapped region. As a light ray will pass through equal volumes of each region (assuming many cells are traversed), the effect of this imperfect matching is expected to be negligible. So, provided many cells are propagated through, τ can be considered an approximate time coordinate throughout the lattice universe, and may be interpreted as the cosmological time.

2.2.1 Interpretation and cosmological significance of the Schwarzschild approximation

Consider again a radially out–falling particle on the surface of an expanding Schwarzschild cell. As discussed in [14, 19], one can think of it as a particle thrown away from the central mass m, with radial coordinate velocity (2.15) and energy $c^2\sqrt{E}$, as measured in the cosmological time coordinate τ. The expansion of the boundary (described by the particle on it) can then be interpreted in terms of the escape velocity $\sqrt{2Gm/(c^2 r)}$. If $E < 1$, then from (2.15), the velocity of the particle is less than the escape velocity. Hence the particle will reach some maximum radius of expansion, before falling back inward. This corresponds to the Schwarzschild cell expanding, and eventually recollapsing. As this holds for every identical cell, and hence the overall lattice, this is analogous to the expansion and contraction of a closed Friedmann universe of positive spatial curvature. For the case $E = 1$, the particle will be travelling at the escape velocity, just high enough to avoid collapse, and in some sense reaching zero velocity at infinity. This corresponds to the expansion of a spatially flat Einstein–de Sitter universe. The last case, for $E > 1$, corresponds to a velocity greater than the escape velocity, such that the particle will always have positive outgoing velocity, even at spatial infinity. This corresponds to an open Friedmann universe of negative spatial curvature.

The most impressive result of the LW model is that this qualitative correspondence between the LW lattice universe and the FLRW one can be made quantitatively explicit. By identifying the radius of the Schwarzschild cell to be the cosmological scale factor, $r = a(\tau)$, then dividing (2.15) by a^2 gives

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{2Gm}{a^3} + \frac{(E-1)c^2}{a^2}\,. \tag{2.17}$$

In the spatially flat case ($E = 1$), this is just the Friedmann equation (2.5) for a density $\rho = m/(\frac{4}{3}\pi a^3)$, as expected for a spherical cell of radius a.

However, this is not the case for a negatively spatially curved space, as the volume of a sphere of radius a is no longer given by $\frac{4}{3}\pi a^3$. Instead, assume that the hyperbolic cube (and hence Schwarzschild cell of radius a) has a volume

$$V_{HC} = \alpha^3 \left(\frac{4}{3}\pi a^3\right)\,. \tag{2.18}$$

Note that, using (2.8) and evaluating the above relation at the present epoch, one finds

$$\alpha = \left(\frac{6V_p}{\pi}\right)^{1/3} \approx 0.79\,. \tag{2.19}$$

The density of the hyperbolic cell (and Schwarzschild cell) is $\rho_{HC} = m/V_{HC}$, so substituting for m in (2.17) with $E \neq 1$, one finds

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{8\pi G\rho_{HC}}{3}\alpha^3 + \frac{(E-1)c^2}{a^2}\,. \tag{2.20}$$

Rescaling all distances such that $a \to \tilde{a} = a\alpha$, and noting that $\tilde{\rho}_{HC} = \rho_{HC}$ and $(\dot{\tilde{a}}/\tilde{a})^2 = (\dot{a}/a)^2$, this becomes

$$\left(\frac{\dot{\tilde{a}}}{\tilde{a}}\right)^2 = \frac{8\pi G\rho_{HC}}{3} + \frac{(E-1)c^2\alpha^2}{\tilde{a}^2}\,, \tag{2.21}$$

which is the usual Friedmann equation, with a few caveats. Firstly, while the dynamics will remain the same, the absolute scale will be different by a factor of α. For example, in a universe of positive spatial curvature, the maximum of expansion will be a factor of α less than that described by the actual Friedmann equation. Secondly, on reformulating this in terms of Ω_{m0} at t_0, one must have

$$\frac{(E-1)c^2\alpha^2}{\tilde{a}_0^2 H_0^2} = 1 - \Omega_{m0}\,, \tag{2.22}$$

to be consistent with the Friedmann equation. This then determines the positive constant E to be

$$E = \frac{(1-\Omega_{m0})\tilde{a}_0^2 H_0^2}{\alpha^2} + 1\,. \tag{2.23}$$

However, this value of $\tilde{a}_0$ should correspond to the a_0 calculated in (2.9), since (2.9) should really be in terms of tilde quantities to describe the Friedmann equation in the LW universe, as opposed to the FLRW one.

To summarise, we have shown that the locally inhomogeneous LW model does have the same dynamical evolution as an FLRW one in a hyperbolic universe, although there is a difference in scales due to the factor α. As discussed in §4.1, this partially motivated the current research, as it gives subtle hints that the hyperbolic LW universe may have optical properties that decrease the need for dark energy. It also leaves open the possibility for future research, since, as far as we are aware, this use of α is original, and may be applied to other models that appear not to take this detail into consideration.

Chapter 3

Optics of a hyperbolic Lindquist–Wheeler universe

As discussed in §2.2.1, the dynamics of the LW universe approaches that of the FLRW one, with subtle differences. However, only recently were the optical properties of such a universe considered by Clifton and Ferreira [19]. Their approach focused on the case of a spatially flat LW universe, whereas the focus here is on a LW universe of constant negative spatial curvature. Most of the nontrivial adjustments necessary were associated with the geometry, and have already been discussed in §2 or Appendix A. Indeed, the theory to consider the propagation of light rays in the LW universe remains largely unchanged from that originally proposed by Clifton and Ferreira, to which the reader is directed for further detail. The following sections are meant to complement the explanation given in [19], and illustrate the changes that occur when moving to a hyperbolic spatial geometry. They are necessarily brief, and hence are not meant to be self–contained, but read in conjunction with [19].

3.1 Light propagation within a cell

In a single cell, the Euler–Lagrange equations may be applied to a space–time geometry specified by the metric (2.14), to determine the geodesic equations for a null light ray, as discussed by Clifton and Ferreira. For use in the numeric code, we work in S.I. units, unlike Clifton and Ferreira who use $c = G = 1$. For the metric (2.14), the corresponding Lagrangian is $L = g_{\mu\nu}\dot{x}^\mu\dot{x}^\nu$, where we have changed our notation such that a dot now represents a derivative with respect to the affine parameter λ. The Euler–Lagrange equations for the given Lagrangian are

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\left((1-\Delta)\,c\dot{\tau} + \dot{r}\sqrt{E-\Delta}\right) = 0 \tag{3.1a}$$

$$\ddot{r} - c\,\ddot{\tau}\sqrt{E-\Delta} \;=\; -\frac{Gm\dot{\tau}^2}{r^2} + Er\left(\dot{\theta}^2 + \sin^2\theta\,\dot{\phi}^2\right) \tag{3.1b}$$

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\left(r^2\theta\right) = r^2\sin\theta\cos\theta\,\dot{\phi}^2 \tag{3.1c}$$

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\left(r^2\sin^2\theta\,\dot{\phi}\right) = 0 \tag{3.1d}$$

which result in three constants of motion B, J, J_ϕ, specified by

$$\dot{\tau} = \frac{EB}{c^2\,\Delta} - \frac{\dot{r}\sqrt{E-\Delta}}{c\Delta} \tag{3.2a}$$

$$\dot{\theta}^2 = \frac{1}{r^4}\left(J^2 - \frac{J_\phi^2}{\sin^2\theta}\right) \tag{3.2b}$$

$$\dot{\phi} = \frac{J_\phi}{r^2\sin^2\theta}\,. \tag{3.2c}$$

The null constraint gives

$$0 = -c^2\Delta\dot{\tau}^2 - 2\,c\sqrt{E-\Delta}\,\dot{\tau}\,\dot{r} + \dot{r}^2 + E\,r^2\left(\dot{\theta}^2 + \dot{\phi}^2\sin^2\theta\right), \tag{3.2d}$$

which follows from the metric (2.14). These equations allow for the calculation of the numerical values of the derivatives of the parameters along the light ray's path, which are then used in a numerical integration procedure to calculate the values of these parameters. In this way, the path of any light ray through a given Schwarzschild cell, and its optical properties, can be determined.

3.1.1 Calculating the redshift

As discussed previously, the optical properties of any universe are extremely important. One significant example is the redshift, and hence in any useful application of the LW universe one should be able to calculate the redshift of a given light ray. Clifton and Ferreira give a simple account of how to do so in the flat case, and this remains unchanged in a hyperbolic universe. To summarise, the redshift z_{LW} in the LW universe is governed by

$$1 + z_{LW} = \frac{\nu_e}{\nu_o}, \tag{3.3}$$

where ν_e is the frequency of the light ray measured at the point of emission, and ν_o is the frequency measured at the point of observation. The frequency of light seen by an observer moving with a four-velocity u^μ is just $\nu = -u^\mu k_\mu$, where

$$k^\mu = \frac{\mathrm{d}x^\mu}{\mathrm{d}\lambda} = \left(\dot{\tau}, \dot{r}, \dot{\theta}, \dot{\phi}\right) \tag{3.4}$$

is the four-vector tangent to the null geodesic of the observed ray. If the observers at the points of emission and observation are to be "comoving", they need to be at the same radial coordinate (given that the scale factor in the LW universe is defined to be the radius of the Schwarzschild boundary). Hence they both must have the same four–velocity u^μ given by (2.16). It is then easily verified that $\nu = -u^\mu k_\mu = c^2\dot{\tau}$, so that the redshift measured by a comoving observer is

$$1 + z_{LW} = \frac{\dot{\tau}_e}{\dot{\tau}_o}\,, \tag{3.5}$$

where $\dot{\tau}_e$ and $\dot{\tau}_o$ are the rate of change of the cosmological time with respect to the affine parameter λ on the null geodesic, at the points of emission and observation, respectively.

These values can be calculated numerically from (3.2a), and hence so can the redshift in the LW universe.

Of course, one must also compare the redshift in the hyperbolic LW universe to what it would be in the hyperbolic FLRW one. To calculate the FLRW redshift z_{FLRW} in a hyperbolic universe, the scale factor a at a given τ must be known. This relation is given by the standard parametric solutions to the Friedmann equation (2.5)

$$a(\eta) = \frac{a_0\Omega_{m0}}{2\left(1-\Omega_{m0}\right)}\left(\cosh(\eta)-1\right), \tag{3.6a}$$

$$\tau(\eta) = \frac{a_0\Omega_{m0}}{2c\left(1-\Omega_{m0}\right)}\left(\sinh(\eta)-\eta\right), \tag{3.6b}$$

where η is the conformal time, and a_0 is the value calculated in (2.9). In general, for a given time τ, (3.6b) may be solved numerically for the conformal time η_τ that corresponds to τ, and then (3.6a) determines $a(\eta_\tau)$. Knowing $a(\tau)$ and a_0 (which is that defined in (2.9)) means the FLRW redshift can then be calculated

$$1 + z_{FLRW} = \frac{a_0}{a(\tau)}. \tag{3.7}$$

It also follows from (3.6a) that

$$\eta_0 = \cosh^{-1}\left(\frac{2-\Omega_{m0}}{\Omega_{m0}}\right), \tag{3.8}$$

and hence a value can be found for $\tau_0 = \tau(\eta_0)$.

3.1.2 Calculating the shear and angular diameter distance

As mentioned in the introduction, the different geometry of the LW and FLRW models will result in different cosmological distance measures in each model. For example, the angular diameter distance can be calculated from the integral of the expansion scalar ϑ along the path of the light ray,

$$r_A \;\propto\; \exp\left(\int_e^o \vartheta \mathrm{d}\lambda\right). \tag{3.9}$$

The evolution of the expansion along a given path is governed by the Sachs optical equations (1.1), which describe the evolution of a bundle of light rays. As discussed in the introduction, in the spatially flat homogeneous FLRW universe, the Ricci tensor is everywhere nonzero, whilst the Weyl curvature is everywhere equal to zero. Correspondingly, there is no shear in an FLRW universe. However, in the LW lattice universe the opposite is true — the Weyl curvature will in general be nonzero, and the Ricci curvature will be zero everywhere outside of the central mass. This results in a nonzero shear in the LW model. These factors clearly alter the evolution of the expansion scalar, as shown in the Sachs equations (1.1), and hence the angular diameter distance (3.9). In general, the nonzero shear acts to make luminous sources appear brighter, whereas the zero Ricci curvature makes them appear dimmer [25].

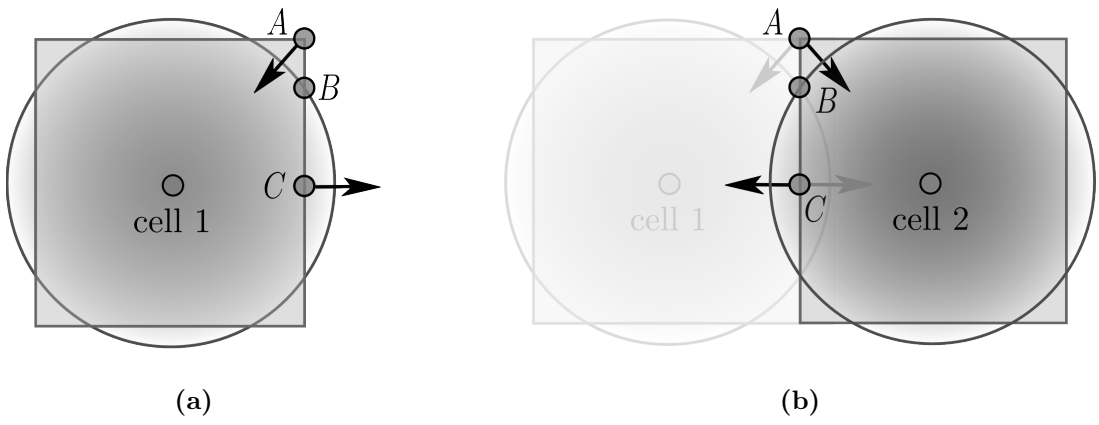
The procedure for calculating the shear and expansion scalars along the path of a light ray is clearly laid out in [19], and requires no adjustment in the case of a hyperbolic universe, and hence is not discussed here.

3.2 Light propagation through cell boundaries

Thus far it has been shown how to determine the path of a light ray within individual LW cells, and how to calculate certain optical properties along that path. To consider the optical properties of an entire LW lattice of such cells there must also be a method for propagating light rays from one cell to another. A brief discussion of how Clifton and Ferreira do so [19] is outlined below, with further details given about the relative motion of observers in differing cells.

3.2.1 Peculiar observers

In general, it may be expected that moving from the coordinates of one cell to those of another adjacent to it would be through a simple translation of the form $x \to x + 2D$, where D is the hypercubic cell dimension. However, as outlined in [19], the condition that the light ray is outgoing in the first cell, but incoming in the second, will violate the null constraint (3.2d), if this method is applied. To arrive at a consistent approach, one makes the reasonable requirement that the frequency of a light ray as measured by two comoving observers should be the same. This has certain nuances that must be considered carefully, as outlined below.



(a) **(b)**

Figure 3.1 – *As discussed in the text, (a) illustrates the peculiar velocities of different observers on the square cell boundary, as seen from cell 1. The peculiar velocity as seen from cell 2 is in the opposing direction (with respect to the horizontal), as shown in (b). This causes opposite redshifts and blueshifts in the overlapping and uncovered regions.*

For simplicity, assume that space is two dimensional and spatially flat, and consider observers A, B, and C on the square cell boundary, as in Fig. 3.1a. As discussed previously, two observers are comoving in the LW sense if they are located at the same radius, and are comoving with respect to the expansion of the whole lattice universe if they are on the circular cell boundary, with radial coordinate a (the scale factor). Hence an observer B located at the intersection of the circular cell boundary and the square one, with radial coordinate a, may be considered comoving with the lattice.

By the same argument, any observer on the boundary not at one of the points B, will have some peculiar velocity with respect to the comoving lattice. For example, an observer A in the uncovered region of tiling has a greater radial coordinate than the circular boundary, and hence by (2.15), has a radially inward peculiar velocity. Similarly, an observer C in the overlap region has a smaller radial coordinate, so has a radially outward peculiar velocity. This is illustrated in Fig. 3.1a.

Now consider these same observers on the boundary, but viewed from the perspective of cell 2. The same arguments as above still hold, and hence the peculiar velocities of the observers from the point of view of cell 2 are different to those of cell 1 (except at points B), as illustrated in Fig. 3.1b. In some sense, the observer at A (C) from the point of view of cell 1 is not comoving with respect to the observer A (C) from the point of view of cell 2. Therefore, in switching between cells, as will happen during light propagation, one must keep in mind that only observers at B can be considered stationary on the boundary.

This becomes a problem when one considers that an observer at a point should observe only one frequency for a light ray. However, due to their differing peculiar velocities in different cells, observers at A or C will observe differing frequencies due to the Doppler effect, depending which cell one is considering them from. Therefore, prescribing that observers on the boundary should measure the same frequency for a light ray crossing the boundary would be incorrect. The correct formulation should be that observers on the boundary should observe the same frequency for a light ray, *when their relative peculiar motions are accounted for*.

Clifton and Ferreira provide an accurate method for calculating this relative peculiar velocity due to changing from one cell to another. However, they also found that the naïve approach of ignoring this relative motion will give the same results, as long as sufficiently many cell crossings are considered. The reason for this is simple — as shown in Fig. 3.1a, the peculiar motion of A is in an opposite direction (with respect to the horizontal) to the peculiar motion of B. Therefore, while one observer will see a Doppler redshifted photon, the other observer will see it blueshifted. Hence, when many cells are considered, and light rays pass through a similar volume of overlap region as uncovered region, the net effect cancels out.

3.2.2 Matching optical parameters

In addition to specifying the observers, as discussed previously, one must also specify how they will match optical the optical parameters $(\dot{\tau}, \dot{r}, \dot{\theta}, \dot{\phi})$ at the boundary. Clifton and Ferreira discuss this in depth in Section III of [19], to which we refer the reader for more detail. However, the following changes should be noted when $c \neq 1$. Firstly, using the fact that $-u^\nu k_\nu = c^2 \dot{\tau}$, the tangent vector k^μ to the light ray may be decomposed as

$$k^\mu = \frac{\mathrm{d}x^\mu}{\mathrm{d}\lambda} = \left(\dot{\tau}, \dot{r}, \dot{\theta}, \dot{\phi}\right) = \left(\frac{-u^\nu k_\nu}{c^2}\right)(u^\mu + n^\nu) \tag{3.10}$$

where u^μ is the four-velocity (2.16), and $n^\mu = c\,(0, n^r, n^\theta, n^\phi)$ is the four-vector orthogonal to u^μ, with magnitude c^2. Hence the equations (28), (29), and (30), of [19] become

$$\dot{\theta} = c\dot{\tau} n^\theta\,, \qquad \dot{\phi} = c\dot{\tau} n^\phi\,, \qquad \dot{r} = c\dot{\tau}\left(\sqrt{E - \Delta} + n^r\right)\,. \tag{3.11}$$

Also, to evolve shear between cells in the general $E \neq 1$ case, the four–vector $\bar{t}^a$ given in (79) of [19] must be changed to

$$\bar{t}^a = \frac{r\dot{\tau}}{\sqrt{2}J}\left[\frac{in^r}{\sqrt{E}},\ i\left(\sqrt{E} + \frac{\sqrt{E - \Delta}\,n^r}{E}\right), -\sin\theta\, n^\theta,\ \frac{n^\phi}{\sin\theta}\right], \tag{3.12}$$

where $\bar{t}^a$ is their notation, and the over–bar does not indicate complex conjugation, though $i = \sqrt{-1}$ (complex numbers are part of the formalism used in constructing the vectors to evolve shear, yet all physical observables are still real). The rest of the results described by Clifton and Ferreira trivially carry over to the case of negative spatial curvature.

3.3 Numerical procedure

As discussed in §4.1, the initial FORTRAN program used by Clifton and Ferreira contained a simple error that resulted in an incorrect implementation of the theoretical procedure. Once this error is corrected, the numerical procedure for a hyperbolic LW universe follows essentially the same principles as for the flat case described by Clifton and Ferreira. The aforementioned changes had to be incorporated, which did result in some nontrivial programming differences. Notably, the initial set–up of the geometry required the development of a numerical integrator, and solving equations (3.6) required the development of a numerical equation solver (which was based on the Newton–Rhapson method).

The present version of the code is given in Appendix B. However, as a majority of it is associated with initialising variables and data storage, the general reader may gain more from a general outline of the procedure followed in the code:

1. for a given mass and relative density Ω_{m0}, the initial properties of the LW space–time are calculated, including a_0, and the size of the hypercubes D_0.

2. using $\dot{\tau}, \dot{r}, \dot{\theta}$, and $\dot{\phi}$ from the first cell (either randomly generated or specified by the user), the constants of motion in (3.2a), (3.2b), and (3.2c) are calculated.

3. the light ray is then propagated through the cell (with optical properties like the redshift in both the LW and FLRW models being calculated simultaneously), until it reaches a boundary of the hypercube.

4. using the values of $\dot{\tau}$, $\dot{r}$, $\dot{\theta}$, and $\dot{\phi}$ at the boundary, the values for n^r, n^θ, and n^ϕ are calculated from (3.11).

5. coordinates (r, θ, ϕ) in the first cell are transformed to $(\hat{r}, \hat{\theta}, \hat{\phi})$ of the new cell by first transforming to Cartesian coordinates, then translating the origin $x \to \hat{x} = x + 2D$, and then transforming to the new spherical coordinates.

6. transform n^a to $n^{\hat{a}}$ in the new coordinates, using the same transformation.

7. given that $\dot{\tau} = \dot{\hat{\tau}}$ and any point on the hypercubic boundary satisfies $r = \hat{r}$, use (3.11) to find $\dot{\hat{r}}, \dot{\hat{\theta}}$, and $\dot{\hat{\phi}}$.

8. repeat again from step 2, until a given redshift (or time) is reached.

Chapter 4

Results

4.1 Optics of a spatially flat Lindquist–Wheeler universe

The most significant outcome of this research was the discovery that the main substantive results of Clifton and Ferreira [19] are incorrect. The theory given by Clifton and Ferreira for considering the optics of a LW universe is correct, though a simple mistake in the program meant the numerical procedure did not correctly reflect the prescribed method.

Notably, recall that in the spatially flat Euclidean case, the boundary of the spherical Schwarzschild cell is defined such that it has the same volume as the original cubic cell. For this Euclidean geometry the size $2D_E$ of the cube in terms of the radius a_E of the Schwarzschild cell is

$$D_E = \left(\frac{\pi}{6}\right)^{\frac{1}{3}} a_E \,. \tag{4.1}$$

The numerical procedure should propagate the light ray through the Schwarzschild cell, keeping track of the Euclidean x, y, z coordinates until one of these equals the D_E defined above. This would represent the light ray being at a boundary, and the program would then implement the matching conditions, and transform to the adjacent cell, and continue on.

The error in the original code of Clifton and Ferreira was simply that the boundary of the cube was mistakenly taken to be the radius a_E. Hence the light ray was propagated until at least one of x, y, z was equal to a_E, as opposed to D_E. When the aforementioned error is corrected, the redshift in the LW universe is in fact the same as that in the FLRW one, as shown in Fig. 4.1. While there may have been scope to investigate this further in detail, there is no incentive to do so, as the result now agrees with the FLRW one.

What may require further investigation is how this correction affects the angular diameter distance in the LW universe. It was found that there was a quantitative change due to the correction, however, it appears that the qualitative properties remain unchanged. This is due to Weyl focusing occurring in the Sachs optical equations (and the corresponding presence of nonzero shear), as opposed to the Ricci focusing which occurs in the FLRW case, as discussed in §1.2. As Clifton and Ferreira concluded, nonzero shear results in an angular diameter distance–redshift relation in the LW universe differs from the FLRW one significantly, and caustics can still form. Clifton and Ferreira also carried out theoretical calculations that estimated the critical redshift at which these effects become significant. Once our numerical implementation is carried out, this should also be considered. For further discussion, see Section V of [19].

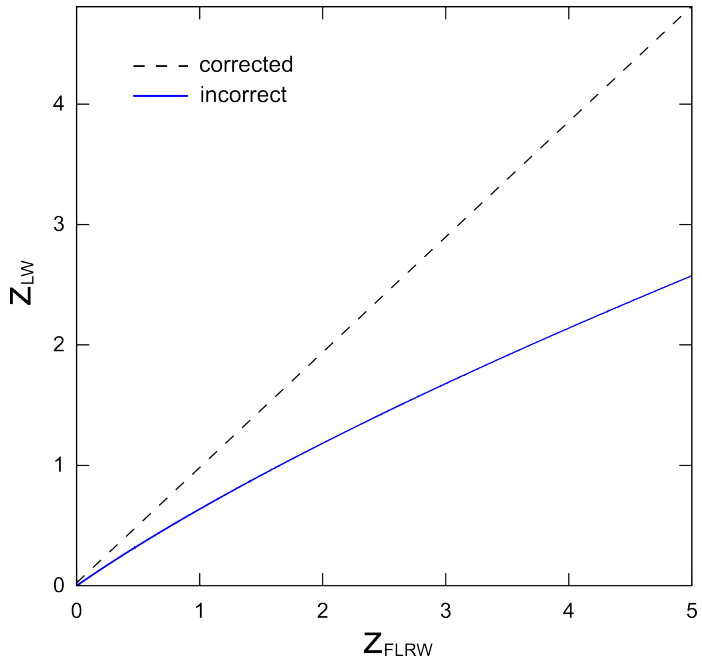


Figure 4.1 – *The correction to the relationship between the redshift z_{LW} in the LW universe and the redshift z_{FLRW} in the FLRW universe.*

While the discovery of the error in the numerical results of Clifton and Ferreira was unexpected, it has a direct impact on the considerations of a hyperbolic LW universe. To see why, note that with this error, the volume of the cell was over estimated, and hence the density under estimated. Due to less gravitationally attractive matter, one would then expect a greater expansion than in a universe with the correct density, and a greater need for dark energy to explain this, as indeed was found in the original results of Clifton and Ferreira. Consider now the hyperbolic universe, where the volume ratio of (2.19) is less than one, and hence the volume is less than that expected in the FLRW model for a given mass, resulting in a greater density. By naïvely applying the reverse of the argument above, one may suspect that a hyperbolic universe will then have a *lesser* need for dark energy. To make a firm prediction, one would have to consider other factors associated with the evolution of a dynamical universe, such as the scale factor in (3.6a). The significance of this prediction seems to justify including this comment, even if it is naïvely motivated.

4.2 Optics of a hyperbolic Lindquist–Wheeler universe

The numerical program I have developed is a modification of the original code of Clifton and Ferreira, that reflects the changes that arise in a hyperbolic universe, as discussed in previous sections. This programming is still proving to be significantly challenging, given I have had no prior programming experience in general, or with the FORTRAN language used by Clifton and Ferreira. Hence, at this stage, a working implementation of the numerical procedure for the case of a LW universe with negative spatial curvature has not been arrived at. This will be the focus of future work.

Chapter 5

Discussion

The dust approximation and the assumption of local homogeneity underpin the standard FLRW cosmology, though their validity is increasingly being questioned [2, 3, 4, 5, 6, 7]. We have considered one method of exploring these assumptions by investigating the properties of the LW discrete lattice cosmology, which is explicitly locally inhomogeneous and discontinuous.

As originally found by Lindquist and Wheeler [14], their universe has similar large–scale dynamics to that of the FLRW one. Notably, one can show that the LW model obeys a dynamical evolution equation of similar form to the standard Friedmann equation. For zero spatial curvature, this is the exact Friedmann equation, though for positive or negative spatial curvature, there is a difference in absolute scale between the equations (which we quantified using the volume scale factor α, in what we believe is an original approach). In any case, the correspondence between the large–scale evolution of the LW and FLRW universe seems to illustrate that the dust approximation appears to be dynamically valid — one can consider the dynamical evolution of an averaged (FLRW) universe to be the same as the average of an explicitly discontinuous (LW) one.

Clifton and Ferreira [19] went further, and considered the optical properties of a spatially flat LW universe. The most significant result of this report is that their principle observationally significant result was found to be incorrect, and the luminosity distance–redshift relation in the LW universe is in fact the same as that in the FLRW one. This means that some of the other observables discussed by Clifton and Ferreira in Section VI of [19] (like the shift parameter, baryon acoustic oscillations, and number density counts) remain the same as in the FLRW case, also in contrast to their original conclusions. Therefore the dust approximation again appears to be valid when considering the redshift and associated optical properties. This may be heuristically understood by noting that the redshift is really just an optical measure of the large–scale expansion history of the universe. Given that this expansion is governed by the dynamical evolution of the universe, one may expect the dust approximation to be valid for redshifts as well, if it is valid for the large–scale dynamical properties of a universe.

Other optical properties of the LW universe were considered by Clifton and Ferreira, and we find that with our correction, these properties remain as claimed, and still differ from those of the FLRW cosmology. This is due to such optical properties depending on the local geometry of the space–time they pass through, which is fundamentally different in the LW and FLRW models. The discrete LW lattice is largely empty space, and hence has Ricci tensor $R_{\mu\nu} = 0$ and a Weyl tensor $C_{\alpha\beta\gamma\delta} \neq 0$. Contrastingly, the perfect fluid assumption

in the FLRW universe results in $R_{\mu\nu} \neq 0$ and $C_{\alpha\beta\gamma\delta} = 0$. As illustrated by (1.1), optical parameters, and the optical characteristics which depend on them, will evolve differently in these two models. Notably, a nonzero shear in the LW model gives rise to the possibility of caustics forming, having a direct effect on the angular diameter distance. Since most of the optical properties depend on the exact local geometry in this way, and not the large–scale average one, the dust approximation is not suitable if one wishes to consider the complete optical properties of a universe.

One such important optical characteristic is the luminosity distance–redshift relation observed for supernovae, which is generally used to support the need for dark energy in the standard FLRW cosmology. Since the luminosity distance–redshift relation was found to be the same in a spatially flat LW universe and an FLRW one, we conclude that the spatially flat LW model provides no alternative explanation for such observations attributed to dark energy. This report considered a LW model of negative spatial curvature, as we believe this may reduce the need for dark energy to some degree. Aspects of hyperbolic geometry were theoretically applied to the LW model, and a numerical method outlined for directly investigating the optical properties of such a universe. Results are yet to be obtained, and will be given in future publications with Clifton and Ferreira.

What of the physical relevance of the corrected LW model? In some sense, this model appears more suitable to describe our current universe than the FLRW one, due to its explicit locally inhomogeneous construction. Correspondingly, the dust approximation is not needed to be made, as the properties of the exact discontinuous dust solution are explored directly, as opposed to the properties of their fluid–like average, which is assumed in FLRW cosmologies. (As discussed above, in some cases the dust approximation does become valid, as this discrete universe has certain properties similar to the continuous FLRW one.) However, the LW model is clearly only a toy model, given that the assumptions of a regular lattice distribution of matter and exact spherical symmetry are unphysical, and not observed in the present universe. One may argue that, in some sense, it is still more physically relevant than the FLRW one — while both cases assume an unphysical distribution of matter, the LW model at least directly incorporates inhomogeneity, while the FLRW does not. This significant difference has physical consequences when certain locally determined optical properties of the universes are considered, such as shear. At this stage, we forgo any further discussion of the averaging problem in cosmology, and refer the reader to [2, 3, 4, 5, 6, 7] as an introduction to why the FLRW model may itself be regarded as an unphysical toy model, and not suitable to describe the universe.

Future work will involve investigating further the optical properties of the LW model (and others such as Swiss cheese ones), in collaboration with Clifton and Ferreira. Notably, we will correct the error made in their original paper, and include my results for the hyperbolic LW universe, after the numerical implementation has been carried out. We suspect that further explorations of the LW model will not be fruitful, as the assumption of spherical symmetry that underpins the LW model is extremely restrictive, and few nontrivial adjustments can be made. One could possibly consider a non–uniform lattice universe (for example, by using a Voronoi tiling, or having different masses in each cell). However, giving up the uniform lattice

means fundamentally new mathematical ideas are required for a successful construction, and they will not just be simple modifications of those used in the LW model. Currently, we see no obvious way toward such a cosmology.

Acknowledgements

As a result of the February 2011 earthquakes in Christchurch, New Zealand, I spent three months on exchange at Oxford University. This was an inspiring opportunity, for which I am extremely grateful. However, it did make continuing this research difficult for a variety of reasons. Most notably, without direct supervision, this work was almost entirely self–directed. I still wish to thank my supervisor, David Wiltshire, for his helpful comments on my return to Christchurch. Furthermore, at Oxford I found many extremely helpful individuals, to whom I would like to extend my gratitude. Specifically, Timothy Clifton and Pedro Ferreira provided valuable insights into their original work, and Krzysztof Bolejko provided further advice, especially with the FORTRAN programming. Lastly, Andrey Astrelin (from Moscow State University) provided many useful comments about the geometric properties of hyperbolic three–dimensional space, via email.

Bibliography

[1] Riess A G, *et al.*1998 *Astronom. J.* **116** 1009

[2] Wiltshire D L 2011 *Class. Quantum Grav.* **28** 164006

[3] Krasiński A *"Inhomogeneous Cosmological Models"* (Cambridge University Press, Cambridge, 1997)

[4] Bolejko K, Célérier M N and Krasiński A 2011 *Class. Quantum Grav.* **28** 164002

[5] Buchert T 2008 *Gen. Relativ. Grav.* **40** 467

[6] Buchert T 2011 *Class. Quantum Grav.* **28** 164007

[7] van den Hoogen R J 2010 arXiv:1003.4020

[8] Lemaître G 1933 *Ann. Soc. Sci. Bruxelles* A **53** 51 [English translation in *Gen. Relativ. Grav.* **29** (1997) 641]

[9] Enqvist K 2008 *Gen. Relativ. Grav.* **40** 451

[10] Szekeres P 1975 *Common. Math. Phys.* **41** 55

[11] Bolejko K and Célérier, M N 2010 *Phys. Rev. D* **82** 103510

[12] Einstein A and Straus E G 1945 *Rev. Mod. Phys.* **17** 120; Err. **18** (1946) 148

[13] Biswas T and Notari A 2008 *J. Cosmol. Astropart. Phys.* JCAP 06 (2008) 021

[14] Lindquist R W and Wheeler J A 1957 *Rev. Mod. Phys.* **29** 432; Err. **31** (1959) 839

[15] Wigner E P and Seitz F 1933 *Phys. Rev.* **43** 804

[16] Wigner E P and Seitz F 1934 *Phys. Rev.* **46** 509

[17] Wiltshire D L 2009 *Int. J. Mod. Phys.* D **18** 2121

[18] Sachs R 1961 *Proc. R. Soc. A* **264** 309

[19] Clifton T and Ferreira P G 2009 *Phys. Rev.* **D**80 103503

[20] Mersini–Houghton L, Wang Y, Mukherjee P, Kafexhiu E 2008 *Astrophys. J.* **29** 167

[21] Coxeter H M S *"Regular Polytopes"* (Methuen and Company Ltd., London, 1948)

[22] Gauss G F, in *"Werke"* (Dieterichschen Universit:ats-Druckerei W. Fr. Kaestner, 1873) pp 175–239 [English translation in Milnor J W, 1982 *Bull. Amer. Math. Soc.* **6** 9]

[23] Gullstrand A 1922 *Ark. Mat. Astron. Fys.* **16** 1

[24] Painlevé P 1921 *C.R. Acad. Sci. (Paris)* **173** 677

[25] Bertotti B 1966 *Proc. R. Soc. A* **294** 195

[26] Press W, Flannery B, Teukolsky S, Vetterling W, *"Numerical recipes in FORTRAN 77: the art of scientific computing, 2nd ed."* (Cambridge University Press, Cambridge, 1992)

Appendix A

Details on tiling a hyperbolic universe

A.1 Coordinates and metrics

A three–dimensional hyperbolic space–like hypersurface of coordinates (x_h, y_h, z_h) can be represented isometrically in a flat four–dimensional embedding space via

$$x_h^2 + y_h^2 + z_h^2 - u^2 = -a^2, \tag{A.1}$$

where u is the embedding coordinate, and a is the radius of embedding such that the constant Gaussian curvature of the spatial hypersurface is $K = -1/a^2$. Through various changes of coordinates, we can arrive at different representations of the hyperbolic spatial hypersurface.

The hyperbolic coordinates (χ, θ, ϕ) are commonly used in cosmology to represent a universe of constant spatial curvature. They can be related to the previous coordinates by

$$x_h = a \sinh\chi \sin\theta \sin\phi \tag{A.2a}$$

$$y_h = a \sinh\chi \sin\theta \cos\phi \tag{A.2b}$$

$$z_h = a \sinh\chi \cos\theta \tag{A.2c}$$

$$u_h = a \cosh\chi\,, \tag{A.2d}$$

where χ parametrises the transformation. The hyperbolic spatial metric $\mathrm{d}\Sigma^2 = \mathrm{d}x_h^2 + \mathrm{d}y_h^2 + \mathrm{d}z_h^2 - \mathrm{d}u^2$ becomes

$$\mathrm{d}s^2 = a^2 \left(\mathrm{d}\chi^2 + r^2 \mathrm{d}\Omega^2\right)\,, \tag{A.3}$$

where $\mathrm{d}\Omega^2 = \mathrm{d}\theta^2 + \sin\theta^2 \mathrm{d}\phi^2$. More importantly, the substitution $r = \sinh\chi$ yields

$$\mathrm{d}s^2 = a^2 \left(\frac{\mathrm{d}r^2}{1+r^2} + r^2 \mathrm{d}\Omega^2\right)\,, \tag{A.4}$$

which is the spatial part of the usual Robertson–Walker metric in hyperbolic space. In this space, the radial proper length is

$$\ell_r = a\, \sinh^{-1}(r)\,. \tag{A.5}$$

The stereographic coordinates (x_p, y_p, z_p) are those in which

$$x_p = \frac{x_h}{u + a} \tag{A.6a}$$

$$y_p = \frac{y_h}{u+a} \tag{A.6b}$$

$$z_p = \frac{z_h}{u+a} \tag{A.6c}$$

$$u = \frac{a(1+r_p^2)}{1-r_p^2}, \tag{A.6d}$$

where $r_p^2 = x_p^2 + y_p^2 + z_p^2$. The hyperbolic spatial metric $\mathrm{d}\Sigma^2 = \mathrm{d}x_h^2 + \mathrm{d}y_h^2 + \mathrm{d}z_h^2 - \mathrm{d}u^2$ becomes

$$\mathrm{d}\Sigma^2 = \frac{4a^2(\mathrm{d}x_p^2 + \mathrm{d}y_p^2 + \mathrm{d}z_p^2)}{(1-r_p^2)^2}\,. \tag{A.7}$$

This forms a representation of the Poincaré ball. The radial proper length in the Poincaré ball is then

$$\ell_{r,p} = 2a\tanh^{-1}(r_p). \tag{A.8}$$

Lastly, we have derived hyperbolic geodesic coordinates (X, Y, Z) on the Poincaré ball — those in which the axes, and lines "parallel" to them, specify geodesics in the Poincaré ball (i.e., circles perpendicular to the boundary).[1] To the best of our knowledge, these to do not exist in the literature. However, they may be derived from the transformation properties of the Poincaré ball, resulting in

$$X = \frac{1 + r_p^2 - \sqrt{(1+r_p^2)^2 - 4x_p^2}}{2x_p} \tag{A.9a}$$

$$Y = \frac{1 + r_p^2 - \sqrt{(1+r_p^2)^2 - 4y_p^2}}{2y_p} \tag{A.9b}$$

$$Z = \frac{1 + r_p^2 - \sqrt{(1+r_p^2)^2 - 4z_p^2}}{2z_p} \tag{A.9c}$$

and are illustrated in Fig. A.1. There appears to be no simple expression for the metric of hyperbolic geodesic coordinates based on the above transformations, so these coordinates are unfortunately of little use in calculating general distances and volumes. Thankfully, the distances along each coordinate axis do coincide with the standard Poincaré ball distances, so the dimensions of the hypercubic will remain the same. Furthermore, as discussed in §2.1.2, these coordinates will be useful as they have the same symmetries as that of spatially flat space Euclidean ones, given that they represent coordinate geodesics.

For all of the given coordinate systems, if isotropy and homogeneity are assumed, then the most general space–*time* is

$$\mathrm{d}s^2 = -c^2\mathrm{d}t^2 + \mathrm{d}\Sigma(t)^2\,, \tag{A.10}$$

where the t dependence of $\mathrm{d}\Sigma$ is due to the assumption that a now depends on t. Then $a(t)$ can be thought of as a conformal scale factor, and the Gaussian curvature of the spatial hypersurface at time t is just $K = -1/a^2(t)$.

[1] Much thanks goes to Andrey Astrelin for his help on developing these coordinates.

A.2 Tiling the Poincaré ball

The hypercubic tiling of three dimensional hyperbolic space we have chosen to consider can be illustrated by a $z_p = 0$ slice representing the Poincaré disk, as in Fig. A.1. The hypersquares shown are described by geodesics in the Poincaré disk, hence appearing concave in this representation. Combined with the requirement that there are five hypersquares around each vertex in Fig. A.1 (such that the interior angles must be 72°), this uniquely specifies the dimension d of the hypersquare. The standard geometry of the Poincaré disk requires geodesic curves to be sections of circles of radius $r > 1$ centred at some distance $a > 1$ from the origin, such that the circle intersects the boundary of the Poincaré disk at right angles, as illustrated in Fig. A.1. The distance $d = a - r$ can then be calculated as follows.

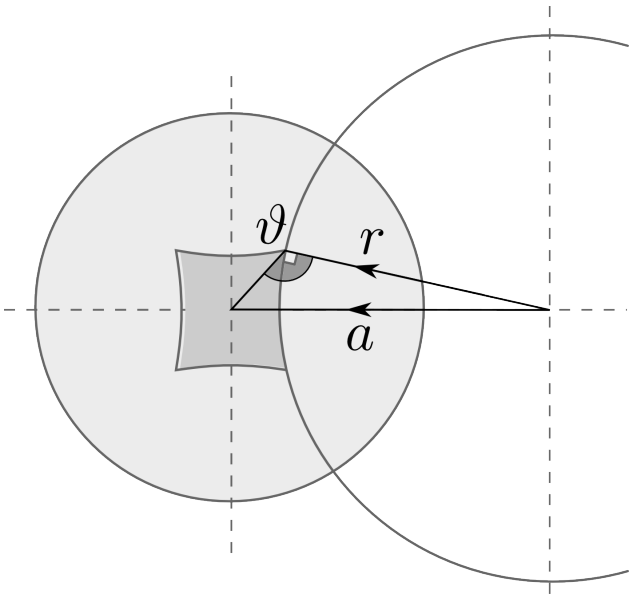


Figure A.1 – *The construction of geodesics and the hypersquare tiles in the two–dimensional Poincaré ball.*

Firstly, the angle ϑ is clearly half of the interior angle of the hypersquare plus a right angle, so is 126°. By the sine rule, it is clear that

$$a = r\sqrt{2}\sin(126^\circ) = r\frac{\sqrt{\sqrt{5}+2}}{\sqrt{\sqrt{5}+1}}, \tag{A.11}$$

where the result $\sin(126^\circ) = (\sqrt{5}+1)/4$ was used. Considering the boundary of the Poincaré disk specified by $x_p^2+y_p^2 = 1$, the slope of the tangent at any point on the boundary is given by $m_1 = -x_p/y_p$. Likewise, for the circle of radius r centred at a, the slope is $m_2 = (a-x_p)/y_p$. Requiring these two circles to intersect perpendicularly is equivalent to requiring $m_1 m_2 = -1$, or $ax_p = x_p^2 + y_p^2 = 1$, and hence $x_p = 1/a$ and $y_p = \sqrt{a^2-1}/a$. Substituting this result, and (A.11) into the formula for the circle of radius r centred at a gives the result $a = \sqrt{\sqrt{5}+2}$ and $r = \sqrt{\sqrt{5}+1}$, and hence the dimension d of the hypersquare in the Poincaré disk (and

the hypercube in the Poincaré ball) to be

$$d = a - r = \sqrt{\sqrt{5} + 2} - \sqrt{\sqrt{5} + 1}\,. \tag{A.12}$$

In the Poincaré disk (and ball), this distance d specifies the unique size of the cells that will completely tile the hyperbolic universe. This d corresponds to the radial proper distance

$$D = \ell_{r,p}(d) = 2a(t)\tanh^{-1}(d)\,. \tag{A.13}$$

A.3 Finding the hypercubic volume

Finding the volume of the hypercube of dimension D is necessary, but considerably more challenging than in the Euclidean case. Given the Poincaré metric (A.7), and corresponding metric determinant

$$\mathrm{g} = \frac{64a^6(t)}{(1 - r_p^2)^6}\,, \tag{A.14}$$

the volume is given by

$$V = 8a^3(t)V_p, \tag{A.15}$$

where V_p is the volume in the Poincaré ball, given by

$$V_p = \int_{-B_{z_p}}^{B_{z_p}} \int_{-B_{y_p}}^{B_{y_p}} \int_{-B_{x_p}}^{B_{x_p}} \frac{\mathrm{d}x_p\,\mathrm{d}y_p\,\mathrm{d}z_p}{(1 - x_p^2 - y_p^2 - z_p^2)^3}, \tag{A.16}$$

and B_{x_p}, B_{y_p}, and B_{z_p} represent the functions specifying the boundary of the hypercube. Determining these boundaries, and implementing the integration with concave boundaries, makes this approach difficult at best. One could instead attempt to work in the hyperbolic geodesic coordinates (A.9), where the boundaries are simply specified by $x = \pm D$, $y = \pm D$, and $z = \pm D$. However, converting the metric into these coordinates is non-trivial.

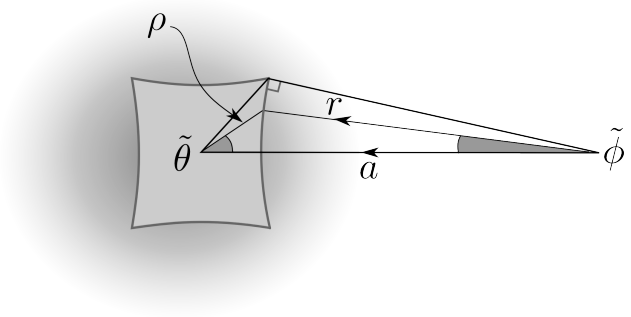


Figure A.2 – *An illustration of variables used to calculate the hypercubic cell volume.*

The solution is to work in spherical coordinates, as illustrated in Fig. A.2. The hypercube is split into six equal parts, each being specified by a $\tilde{\theta}$ and $\tilde{\phi}$ coordinate range of $\pi/2$. The

radius ρ must be specified as a function of $\tilde{\theta}$ and $\tilde{\phi}$ to carry out the integration. Considering the positive x face, the bounding surface is specified by

$$(x_p - a)^2 + y_p^2 + z_p^2 = r^2, \tag{A.17}$$

where a and r are those already calculated. In spherical coordinates with $x_p = \rho \sin(\tilde{\theta}) \cos(\tilde{\phi})$, where $\rho^2 = x_p^2 + y_p^2 + z_p^2$, this becomes

$$\rho^2 - 2a\rho \sin(\tilde{\theta}) \cos(\tilde{\phi}) + a^2 = r^2, \tag{A.18}$$

which has solution

$$\rho = a \sin(\tilde{\theta}) \cos(\tilde{\phi}) - \sqrt{a^2 \sin^2(\tilde{\theta}) \cos^2(\tilde{\phi}) - a^2 + r^2}. \tag{A.19}$$

Converting the metric into spherical coordinates, and the corresponding volume element, then (A.16) becomes

$$V_p = \int_{\pi/4}^{3\pi/4} \int_{-\pi/4}^{\pi/4} \int_0^{\rho(\tilde{\theta},\tilde{\phi})} \frac{r^2 \sin(\tilde{\theta}) \mathrm{d}\tilde{r} \mathrm{d}\tilde{\phi} \mathrm{d}\tilde{\theta}}{(1-\tilde{r}^2)^3} \tag{A.20}$$

$$= \frac{1}{8} \int_{\pi/4}^{3\pi/4} \int_{-\pi/4}^{\pi/4} \int_0^{\rho(\tilde{\theta},\tilde{\phi})} \left[\frac{(\tilde{r}^2+1)\sin(\tilde{\theta})\tilde{r}}{(1-\tilde{r}^2)^2} - \sin(\tilde{\theta}) \tanh^{-1}(\tilde{r}) \right] \mathrm{d}\tilde{r} \mathrm{d}\tilde{\phi} \mathrm{d}\tilde{\theta}, \tag{A.21}$$

where $\tilde{r}$ is just a dummy variable. Analytically it appears one can proceed no further, Nonetheless, given that the region of integration in now convex, and all the boundaries are well-defined, the integral can be solved numerically. A program in FORTRAN was developed to do this integral. This gives the volume element in the Poincaré ball $V_p \approx 0.289$.

Appendix B

Numerical code

This section gives the most recent version of the numerical code being used to model the optical properties of a hyperbolic LW universe. We have used the first method for matching between cells, as specified in [19]. This makes the code simpler, and according to the results of Clifton and Ferreira, it should give the same results as the more accurate method, in the regime we are considering.

The code below has been briefly annotated so that the reader may gain an understanding of the processes involved. Detail has been intentionally left out in certain areas, as we feel the reader must develop their own understanding of the code if they wish to work with it, without making trivial errors. These areas include many of the subroutines and functions, most notably those associated with the propagation of shear, and also the numerical integrators and solvers. For more information on the latter, the reader is referred to [26], from which these parts of the code were based.

B.1 Glossary

There are many variables utilised in the coder, and we hope this quick reference will make it more approachable for the reader.

`a1` = scale factor at `tau1`
`ai` = initial scale factor in first cell
`alphaI` = imaginary component of `alpha` used in angular diameter calculations
`alphaR` = real component of `alpha` used in angular diameter calculations
`alpha` = volume ratio factor
`a` = scale factor
`B,J,Jph` = integration constants
`cc` = speed of light
`cellsize` = size of the cell boundary
`const1` = constant used in parametric equations for `a` and `tau`
`c` = vector for use in `dverk`
`dboundeta` = bound on `eta`
`dd` = variable for subroutine `calcalphappsi`
`dellambda = lambdamax/float(nbins-1)` is increment of `lambda`. Note, this is negative, as we are going back in time.
`delzFRW = redshifti/(nbins-1)`

`delz = (redshiftlim-redshiftmin)/dfloat(nbins-1)` = increment from `redshiftmin` to `redshiftlim` such that there are `nbins` increments.
`dlarray` is the luminosity distance. Each `dlarray(i)` gives the total luminosity distance measured across the cells that correspond to a given `i`.
`dlarraymc` = computes the average luminosity distance passed through in each cell, for the group of cells with the same `i` value (redshift bin).
`dlav(jmc)` = the sum of the luminosity distances per cell, for each redshift bin, over all the `mc` runs carried out. Note that the average luminosity distance passed through per cell, over all the `mc` runs carried out is `dlav(i)/narraymc(i)`
`dotnorm` = norm of vector
`dotrai` = initial rate of change of `ra`
`dotra` = rate of change of `ra`
`dotr, dottheta, dotphi` = velocities — spherical polars
`dotri, dotthetai, dotphii` = velocities in first cell before being propagated — spherical
`dottaui` = rate of change of `tau` with respect to geodesic parameter, in first cell
`dottauo` = initial value for `dottau` (`= dottaui` to start)
`dottau` = rate of change of `tau` with respect to geodesic parameter
`dotxi, dotyi, dotzi` = velocities in first cell before being propagated — hyperbolic geodesic coordinates
`dtbase` = `-dtbaseinp` to specify going back in time
`dtbaseinp` = base input time step
`dt` = `dtbase`, the increment of geodesic parameter
`epsb` = precision in boundary crossing
`epstau` = precision in final time
`eta0` = conformal time at the present epoch
`eta1` = conformal time to integrate back to
`eta` = conformal time
`eta_est` = estimate of `eta` value at each iteration of numerical solver
`eta_max` = minimum bound on value for `eta` calulated by numeric solver
`eta_min` = minimum bound on value for `eta` calulated by numeric solver
`E` = the parameter `E` used by Clifton and Ferreira
`G` = Newton's gravitational constant `h` = as in Hubble `h = 0.67`
`icc` = integer that keeps count of number of iterations of integrator. Initially `icc=0`, then `icc=icc+1 etc.`
`idum` = number for random number generator (note must be negative to initialize, and only need one seed — it regenerates new one each time)
`ilammaxmax` = max of `ilammax` reached
`ilammaxmc` = `ilammax` at end of integration through cells
`ilammax` = number of `dellambda` increments of `xb` above zero
`imatch` = indicator for using different matching condition
`imc` = variable for `do` statements, represents the number of the simulation that is being run (i.e., `i=1,nummc`)

ind = indicator for dverk
iout1 = internal output indicator
iout = internal output indicator
iread = indicator to read initial conditions from file ICPosVel
itrans = transition output indicator
i = variable for do statements
iZFRWmax = as for izmax, but for zFRW
izFRWmaxmax = as for izmaxmax, but for zFRW (note, don't need min, as redshiftmin=0, and then increases).
iZFRWmaxmc = as for izmaxmc, but for zFRW
izmax = maximum i value reached , corresponding to maximum redshift reached. Start izmax=1, so that can use condition if zmax<i then zmax=i.
izmaxmax = maximum of izmax reached. Start =1
izmaxmc = gives izmax for each mc run.
izminmc = gives izmin for each mc run.
izmin = minimum i value reached , corresponding to minimum redshift reached. Maximum value of i occurs when redshift=redshiftlim, which means that i=(redshiftlim-redshiftmin)/delz+1=nbins-1+1=nbins. Hence start izmin=nbins, so that can use the condition if zmin>i then zmin=i.
izminmin = minimum of izmin reached. Start =nbins
jmc = variable for do statements within each simulation, up to nbins (i.e. jmc=1,nbins usually)
lambdai = value of lambda in first cell
lambdamax = -6000
m = central mass of each cell
minput = m in units of Milky–ways
narray = gives the number of cells in which the redshift was measured to be such corresponding to i (note each i corresponds to a delz interval).
narraymc(jmc) = the number of mc runs that measured a redshift in that bin
nbins = number of data points to take for optical variables
ncart(3) = spatial normal vector in old cell — hyperbolic geodesic coordinates
n_eq = number of equations to integrate
nsnew(3) = spatial normal vector in new cell — spherical polars
nsold(3) = spatial normal vector in old cell — spherical polars
nummc = number of simulations to run each time program is run = 1 as read in inputdata, so only one simulation per program run
nummctot = the number of simulations to be run
numout = number of output files
nw=n_eq = number of equations to integrate
nzarray = gives the number of cells in which the redshift was measured to be such corresponding to i (note each i corresponds to a dellambda interval).
nzarraymc = the number of mc runs that measured a redshift in that bin (note each i cor-

responds to a `dellambda` interval)
`nzFRWarraymc` = the number of `mc` runs that measured a FRW redshift in that bin (note each `i` corresponds to a `delzFRW` interval)
`nzredFRWarray(i)` = number of cells in which the FRW redshift is measured in the bin `i` of zFRW
`omega0` = the matter density parameter
`pi` = 3.14159...
`pphi` = the Φ of CF
`pphii` = `pphi` in first cell
`ppsi` = the Ψ of CF
`ppsi0` = `ppsi` in first cell, calculated via `calcalphappsi`
`r1` = a variable assigned to `ran1` — a random integer number between between -1 and 1
`ra` = angular diameter distance
`rai` = initial angular diameter distance
`ran1` = random integer number between 0 and 1, output by function `ran1`, based on the seed `idum`.
`redarray` =the sum of the redshifts that fall in a given bin `i` of `lambda`
`reddav` = the sum of the average redshift per cell, for each redshift lambda bin, over all the `mc` runs carried out. Note that the average redshift passed through per bin, over all the `mc` runs carried out is `reddav(i)/nzarraymc(i)`
`redFRWarray(i)` = the sum of the FRW redshifts that fall in a given bin `i` of `zFRW`
`redFRWdav` = the sum of the average FRW redshift per cell, for each redshift bin, over all the `mc` runs carried out. Note that the average FRW redshift passed through per cell, over all the `mc` runs carried out is `redFRWdav(i)/nzFRWarraymc(i)`
`redshiftFRW` = FRW redshift
`redshiftFRWmc` = computes the average FRW redshift passed through in each cell, for the group of cells with the same `i` value (FRW redshift bin)
`redshifti` = final redshift to integrate back to
`redshiftlim` = `redshifti` = maximum redshift to be considered.
`redshiftmc` = computes the average redshift passed through in each cell, for the group of cells with the same `i` value (lambda bin).
`redshiftmin` = `-redshifti` = -final redshift
`redshift` = the LW redshift
`rg` = galaxy radius in Mpc
`ri, thetai, phii` = coordinates in first cell before being propagated — spherical polars
`rsc` = closest radius allowed
`r,theta,phi,tau` = coordinates in cell — spherical polars
`sigmai` = initial shear in first cell
`sigma` = X of CF
`tau0` = tau at the present epoch
`tau1` = time to integrate back to
`tauarray(i)` = `tauarray(i)+tau` is the total tau that is passed through in a given bin `i` of

lambda
`tauav` = the sum of tau per cell, for each lambda bin, over all the `mc` runs carried out. Note that the average luminosity distance passed through per cell, over all the `mc` runs carried out is `tauav(i)/nzarraymc(i)`
`tauf` = final time to be integrated back to
`tau_func` = function to calculate the value of `tau`, and derivative. Used in numerical solver.
`taumc` = computes the average `tau` passed through in each cell, for the group of cells with the same `i` value (lambda bin).
`tcart(4,2)` = variable in program from subroutine `createtb`.
`tcartb(4,2)` = vector used for calculating shear at boundary (`tcartb(i,1)` = real components, `tcartb(i,2)`=complex components). Used in subroutine `createtb` — hyperbolic geodesic coordinates
`ti` = initial geodesic time
`tol1` = `tol` = precision of integrator `dverk`
`tol` = precision of integrator `dverk`
`Vp` = volume of cell in Poincaré ball
`w` = vector for use in `dverk`
`x1,x2,x3` = spatial coordinates after integration by `dt` — hyperbolic geodesic coordinates
`xa` = total geodesic parameter at beginning of integration
`xb=xa+dt` = geodesic parameter at the end of integration by `dt`
`xci,yci,zci` = initial fraction of comoving `cellzise` (-1 to 1), in hyperbolic geodesic coordinates
`xi, yi, zi, taui` = coordinates in first cell before being propagated - hyperbolic geodesic coordinates
`y` = array containing values for `dverk`
`ytemp` = array containing previous iteration of `y` values as a backup (if `y` are unsatisfactory after running `dverk`, then reset to `ytemp`)

B.2 Program code

The abridged version of the code is given in the following pages.

```
        Lattice

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     1. Declaring variables
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



      integer nbins,nummc,nummctot,n_eq,nw,nw1,ind,i,imc,jmc,iout,itrans,
     & icc,idum,iout1,iread,izmax,izmaxmc(nummctot),izmaxmax,izminmin,
     & izmin,izminmc(nummctot),izFRWmax,izFRWmaxmc(nummctot),
     & izFRWmaxmax,ilammax,ilammaxmc(nummctot),ilammaxmax,imatch,numout

      parameter (nbins=200,nummctot=40,n_eq=10,nw=n_eq)

      real*8 tol,tol1,ran1,r1,c(nw+30),w(nw,18),dt,dtbase,dtbaseinp,h,
     & minput,y(n_eq),ytemp(n_eq),ti,xi,yi,zi,taui,dotxi,dotyi,dotzi,
     & tauf,xci,yci,zci,cellsize,dotx,doty,dotz,ra,dotra,sigma,rai,dotrai,
     & sigmai,dotr,dottheta,dotphi,redshift,pphi,
     & pphii,a,epstau,ai,epsb,dottau,redshiftFRW,dotnorm,ri,thetai,phii,
     & dotri,dotthetai,dotphii,dottaui,rsc,r,theta,phi,tau,x1,x2,x3,dottauo,
     & rg,xa,xb,nsold(3),nsnew(3),ncart(3),tsph(4,2),tcartb(4,2),
     & tcart(4,2),narray(nbins),dlarray(nbins),redshifti,delz,redshiftmin,
     & narraymc(nbins),dlav(nbins),dlarraymc(nummctot,nbins),
     & nzarray(nbins),redarray(nbins),lambdamax,dellambda,nzarraymc(nbins),
     & reddav(nbins),redshiftmc(nummctot,nbins),nzFRWarray(nbins),
     & redFRWarray(nbins),nzFRWarraymc(nbins),redFRWdav(nbins),
     & redshiftFRWmc(nummctot,nbins),delzFRW,redshiftlim,tauarray(nbins),
     & tauav(nbins),taumc(nummctot,nbins),E,m,B,J,Jphi,tau0,ppsi0,
     & lambdai,sigma0,alphaR,alphaI,X,dd,a0,G,eta0,eta1,tau1,a1,tau_const,
     & eta_min,eta_max,tau_func,eta_est,eta,dboundeta,pi,cc,omega0,Vp,
     & const1,alpha

c     defining the common block
             /params/ E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,alphaR,
     & alphaI,G,cc

c     external programs/functions to be able to be used as arguments
               derivs,ran1,tau_func,eta_est

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     2. Initializing the parameters
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c     Following opens file inputdata.dat and references it as unit 11
          (11,    ="inputdata.dat",      ="unknown")
          (11,*)ti
          (11,*)dtbaseinp
          (11,*)xci
          (11,*)yci
          (11,*)zci
          (11,*)redshifti
          (11,*)tol
          (11,*)epstau
          (11,*)epsb
          (11,*)minput
          (11,*)h
          (11,*)rg
          (11,*)idum
          (11,*)iout
          (11,*)itrans
          (11,*)nummc
          (11,*)iout1
          (11,*)iread
          (11,*)imatch
          (11,*)omega0
          (11,*)G
```

```
            (11,*)cc
            (11,*)dboundeta
70
             (11)

      c     If iread = 1 then opens file ICPosVeldat to be read later
        (iread    1)    (12,    ='ICPosVel.dat',      ='unknown')

      c     Initialize average dl arrays to zero
            jmc=1,nbins
            narraymc(jmc)=0d0
            dlav(jmc)=0d0
80             imc=1,nummc
               dlarraymc(imc,jmc)=0d0



      c     Initialiaze average redshift arrays to zero
            jmc=1,nbins
            nzarraymc(jmc)=0d0
            reddav(jmc)=0d0
            tauav(jmc)=0d0
90             imc=1,nummc
               redshiftmc(imc,jmc)=0d0
               taumc(imc,jmc)=0d0



      c     Initialiaze average FRW redshift arrays to zero
            jmc=1,nbins
            nzFRWarraymc(jmc)=0d0
            redFRWdav(jmc)=0d0
100            imc=1,nummc
               redshiftFRWmc(imc,jmc)=0d0



      c     Initialize increments.
            redshiftlim=redshifti
            redshiftmin=-redshifti
            delz=(redshiftlim-redshiftmin)/dfloat(nbins-1)
            delzFRW=redshifti/dfloat(nbins-1)
110         lambdamax=-6000d0
            dellambda=lambdamax/float(nbins-1)

      c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      c     3. Begin propagation through lattice universe, for each imc.
      c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      c     for each complete run:
            imc=1,nummc

120   c     icc keeps count of integrations carried out
            icc=0

      c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      c     3.1 Set up first cell
      c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      c     Initial values for light ray
        (iread    1)     !read ICPosVeldat for initial values
             (12,*)xci,yci,zci
130          (12,*)dotxi,dotyi,dotzi
            (iread    1)     !randomize positions and initial velocities.
            r1=ran1(idum)
            xci=(r1-0.5d0)*2d0
            r1=ran1(idum)
```

```
            yci=(r1-0.5d0)*2d0
            r1=ran1(idum)
            zci=(r1-0.5d0)*2d0

            r1=ran1(idum)
            dotxi=(r1-0.5d0)*2d0
            r1=ran1(idum)
            dotyi=(r1-0.5d0)*2d0
            r1=ran1(idum)
            dotzi=(r1-0.5d0)*2d0


      c     Normalise the velocities
            dotnorm=dotxi**2d0+dotyi**2d0+dotzi**2d0
            dotnorm=dsqrt(dotnorm)
            dotxi=cc*dotxi/dotnorm
            dotyi=cc*dotyi/dotnorm
            dotzi=cc*dotzi/dotnorm

      c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      c     3.2 calculate the volume and a0
      c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            m=minput*1d42 !note, the mass of Milky way is taken a 1d42 kg
            H0=h/3.0856d17 !in units of 1/s

      c     calculate the volume in the poincare ball
                 hvolcalc(hvolp)
            Vp=hvolp

      c     calculate alpha
            pi=acos(-1d0)
            alpha=(6d0*Vp/pi)**(1d0/3d0)

      c     set up rg
            rg=rg*3.0856d22

      c     calculate a0
            a0=(pi*G*m/(3d0*omega0*H0**2d0*Vp))**(1d0/3d0)

      c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      c     3.3 calculate time parameters
      c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      c     calculate E
            E=a0**2d0*H0**2d0*(1d0-omega0)/cc**2d0/alpha**2d0+1d0

      c     cell dimension in poincare ball
            cellsize_p=sqrt(sqrt(5d0)+2d0)-sqrt(sqrt(5d0)+1d0)

      c     cell size in universe. Note, this is D/a0...we don't include the factor of a(t),
      c     as the general expression for the cellsize is D proportional to a(t), where
      c     the constant of proportionality is the "cellsize" given below. This way, the
      actual
      c     dimension of the cell is a(t)*cellsize at all times.
            cellsize=2d0*a0*atanh(cellsize_p)

      c     find initial parameters now
            eta0=dacosh(2d0*(1d0-omega0)/omega0+1d0)
            const1=a0*omega0/2d0/(1d0-omega0)
            tau0=const1/cc*(dsinh(eta0)-eta0)

      c     Setting the tau and eta coordinate in the first cell equal to the current time
      (age of universe) tau0
            taui=tau0
            eta=eta0
```

```
c     Determining the final parameters to be integrated back to. Note tauf<tau0...we
are going back in time.
      a1=a0/(1d0+redshifti)
      eta1=dacosh(a1/const1+1d0)
      tau1=const1/cc*(dsinh(eta1)-eta1)

c     setting final time to integrate back to
      tauf=tau1

c     Defining scale factor in first cell to be that today a0
      ai=a0
      a=ai

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     3.4  calculate intial position and velocities
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     Calculating initial coordinates in first cell.
      xi=xci*cellsize
      yi=yci*cellsize
      zi=zci*cellsize

c     Convert coordinates in first cell to polar
           cart2sphe(xi,yi,zi,ri,thetai,phii)
c     Calculates spherical polar velocities dotri, dotthetai, dotphii in first cell,
from
c     cartesian ones dotxi, dotyi, dotzi (and the spherical coordinates)
           vcart2vsphe(ri,thetai,phii,dotxi,dotyi,dotzi,dotri,dotthetai,
     &     dotphii)


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     3.5 calculate the initial values to go into integrator
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c     Calculate the constants of motions in the first cell from the above
c     and a value for dottaui.
           integrals(ri,thetai,phii,dotri,dotthetai,dotphii,dottaui)


c     dottauo = beginning value for dottau (= dottaui to start)
      dottauo=dottaui
      dtbase=-dtbaseinp
      dt=dtbase

c     xa is the initial time increment.  Note, dt<0, so xa=0-dt>0 in first
c     cell.  We then integrate through time dt<0, such that x parameter at end
c     of integration is zero.
      xa=ti-dt

c     Initialize variables to be input into the dverk integrator.
      y(1)=ri
      y(2)=thetai
      y(3)=phii
      y(4)=taui
      y(5)=dotri
      y(6)=dotthetai
      y(7)=sigmai
      y(8)=rai
      y(9)=dotrai
      y(10)=pphii

      a=a0
      eta=eta0

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     3.6 Initialize internal arrays to zero.
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
            i=1,nbins
            dlarray(i)=0d0
            narray(i)=0d0

270
          izmax=1
          izmin=nbins
          iZFRWmax=1

            i=1,nbins
            tauarray(i)=0d0
            redarray(i)=0d0
            redFRWarray(i)=0d0
            nzarray(i)=0d0
280         nzFRWarray(i)=0d0


          ilammax=1

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     3.7 Initial conditions for lattice and optical variables
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

          rai=1d-5
290       sigmai=1d-12
          dotrai=-1d0
          rsc=1d10
          pphii=0d0
          lambdai=0d0
          alphaR=0d0
          alphaI=0d0

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     4. Integrating null geodesic through cell
300   c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c     300 is the restart point for integration
       300

c     increase geodesic parameter after last integration
          xa=xa+dt

c     from 400 on is the integration loop for one time interval dt
       400
310
c     saves a copy of the last y values as ytemp
            i=1,n_eq
            ytemp(i)=y(i)


c     if dtau is too small, then stop.
            (dabs(dt)    1d-30)
                  (6,*)"dt too small,taui,tauf,tau",taui,tauf,tau
                  (10+10*imc+2,*)"Integration of run imc = ",imc,
320       &    " stopped as dt = ",dt," was too small."



c     The actual integration: this uses dverk to integrate by dt.  Note that
c     "derivs" gives the y_dot values.
               integrator(n_eq,derivs,xa,y,dt,tol,ind,c,nw,w)

c     Update xb, which gives total time integrated over. Note xb=xa+dt now.
          xb=xa+dt
330
c     Now redefine the following variables as the ones that have already been
```

```
integrated previously.
      tau=y(4)
      r=y(1)
      theta=y(2)
      phi=y(3)
      dotr=y(5)
      dottheta=y(6)
      dotphi=Jphi/r**2d0/(dsin(theta))**2d0/m
      sigma=y(7)
      ra=y(8)
      dotra=y(9)
      pphi=y(10)

c     Now use these walue of dot...to calculate the new dottau
           integrals(r,theta,phi,dotr,dottheta,dotphi,dottau)

c     And convert the polar coordinates to cartesian x1,x2,x3.
           sphe2cart(r,theta,phi,x1,x2,x3)

c     Now calculate the redshift before integration.


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     4.1 Check if in galaxy
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c     if within a galaxy
        (r    rg)
                  (6,*)imc," in Gal"
           600 !600 is the end of the integration loop


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     4.2 Check if we tau is less than initial FRW tau.  Note the initial tau
c     is greater than the final tauf. Hence if tau is less than tauf (to given
c     precision of epstau), we have gone too far. Hence repeat, but with dt
c     smaller. But if is within epstau of final value, then we're done
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c     if tau is beyond tauf, to precision of epstau
        (tau    (1d0-epstau)*tauf)
c        Make dt smaller before redoing the integration
         dt=dt/2d0
c        Reset y(i)
            i=1,n_eq
            y(i)=ytemp(i)

c        400 begins the integration loop again
              400

c     However if tau<tauf, but > than lower limit set by precision epstau, then we're
      done
           (tau    tauf     tau    (1d0-epstau)*tauf)     !
             integrals(r,theta,phi,dotr,dottheta,dotphi,dottau)
              600


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     5. At this stage, we know that we are not too close to
c     the galaxy, and that we are not at the end of the integration (i.e.
c     tauf).  Hence we need to check whether we have reached a boundary.
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c     Recalculate the scale factor and hence the radius of the new boundary.
      eta_min=eta-dboundeta*eta
      eta_max=eta+dboundeta*eta
           find_a_from_tau(const1,cc,tau,eta_min,eta_max,a,eta)
```

```
            (ISNAN(eta))
                  (6,*)"error in numerical calculation of eta"
                  600
400
             cellsize=a*atanh(cellsize_p)

      c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      c      5.1 If we are outside boundary (with precision of epsb) then make
      c      dt smaller before repeating
      c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

               ((dabs(x1)    (1d0+epsb)*cellsize)
            &     (dabs(x2)    (1d0+epsb)*cellsize)
410         &     (dabs(x3)    (1d0+epsb)*cellsize))
                 dt=dt/2d0

      c          Reset y(i)
                    i=1,n_eq
                    y(i)=ytemp(i)

                      400

      c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
420   c      5.2 If a boundary is crossed, then transfer between cells.  Note, the
      c      cases of crossing different boundaries are treated separately. I have
      c      explicitly included the procedure for crossing the x boundary, and the
      c      others are similar, so I have not included them (note, one must also
      c      check if the x- and y-boundary intersection is crossed, etc).
      c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      c     Check to see if we cross the x face
                  ((dabs(x1)    (1d0+epsb)*cellsize)
            &       (dabs(x1)    cellsize)
430         &       (dabs(x2)    cellsize)
            &       (dabs(x3)    cellsize))

      c      If we do, then apply matching conditions
      c      A. Evaluate the integration constants J, Jphi, B and dottau at the
      c      boundary.
                  integrals(r,theta,phi,dotr,dottheta,dotphi,dottau)

      c      B. Then take this dottau, and already calculated dotr, dottheta,
      c      dotphi and use them to find the vector nsold, normal to the old boundary,
440   c      in spherical polar coordinates.  Specifically n^r=nsold(1),
      c      n^theta=nsold(2), n^phi=nsold(3)
                  vsphe2vnorm(r,theta,phi,dotr,dottheta,dotphi,dottau,nsold)

      c      C. Convert this spherical polar nsold to cartesian ncart.
                  vnorm2vcart(r,theta,phi,nsold,ncart)

      c      D. Create the vector t used for considering shear between cells.
                  createtb(xb,r,theta,phi,dotr,dottheta,dotphi,
            &     dottau,tcart)
450
      c      E. Translate to cartesian coordinates of new cell.
             x1=-sign(1d0,x1)*(2d0*cellsize-dabs(x1))

      c      F. Convert to spherical polar coordinates
                  cart2sphe(x1,x2,x3,r,theta,phi)

      c      G. Convert ncart into the new spherical polar coordinates, to get nsnew.
                  vcart2vnorm(r,theta,phi,ncart,nsnew)

460   c      H. Use the matching condition to calculate dottau, by method 1 only
      c      in this case.
               (imatch    1)     rentaudot(dl2,dotb,r,dotr,nsnew,dottau)
```

```
c     I. Use the dottau (calculated by either method) to calculate dotr,
c     dottheta, and dotphi from dottau and nsnew
           vnorm2vsphe(r,theta,phi,dotr,dottheta,dotphi,dottau,nsnew)

c     J. Recalculate constants of motion in these coordinates.
           integrals(r,theta,phi,dotr,dottheta,dotphi,dottau)

c     K. Calculate alpha and ppsi in new cell
           calcalphappsi(xb,r,theta,phi,dotr,dottheta,
     &     dotphi,dottau,tcart)

c     L. Redefine y, and dt, for the integration through the new cell. dt is
c     reset to the original value (to account for possible dt=dt/2 that may
c     have occured).
      dt=dtbase

      y(1)=r
      y(2)=theta
      y(3)=phi
      y(4)=tau
      y(5)=dotr
      y(6)=dottheta
      y(7)=sigma
      y(8)=ra
      y(9)=dotra
      y(10)=pphi

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     This represents the end of the boundary crossing (assuming all other
c     boundaries have been checked.
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 302

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     6. After transferring to new cell, calculate new optical variables etc
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c     Reset closest approach radius (if applicable)
        (r/2d0/m    rsc)rsc=r*cc**2d0/2d0/m/G

c     Reset dt
      dt=dtbase

c     Add to cell counter
      icc=icc+1

c     Calculate eta and the redshifts
      eta_min=eta-dboundeta*eta
      eta_max=eta+dboundeta*eta
           find_a_from_tau(const1,cc,tau,eta_min,eta_max,a,eta)
      redshiftFRW=a0/a-1d0
      redshift=dottau/dottauo-1d0

c     Figure out which redshift bin the redshift should be placed.
      i=int((redshiftFRW-redshiftmin)/delz)+1!CCHANGE

c     Reset izmin and izmax if applicable, and add to cumulative luminosity
c     distance dl, and to narray (which counts number in this bin)
        (i    izmin)izmin=i
      dlarray(i)=dlarray(i)+(1d0+redshift)**2d0*ra
      narray(i)=narray(i)+1
        (i    izmax)izmax=i

c     Figure out which FRW redshift bin the FRW redshift should be placed.
      i=int(redshiftFRW/delzFRW)+1
```

```
c     Reset izFRWmax if applicable, and add to cumulative redshiftFRW, and to
c     nzFRWarray (which counts number in this bin)
      redFRWarray(i)=redFRWarray(i)+redshiftFRW
      nzFRWarray(i)=nzFRWarray(i)+1
        (i    izFRWmax)izFRWmax=i

c     Figure out which lambda bin the lambda should be placed.
      i=int(xb/dellambda)+1

c     Reset ilammax if applicable, and add to cumulative redshift and tau, and
c     the counter nzarray
      redarray(i)=redarray(i)+redshift
      tauarray(i)=tauarray(i)+tau
      nzarray(i)=nzarray(i)+1
        (i    ilammax)ilammax=i

c     Now all is done for this old cell, and go to 300 to start integration
c     through next cell.
           300

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     7. We are now finished the integrating, as we are within epstau of tauf.
c     We now process the data of the given imc run.
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 600

c     Set transfer the relevant variables for the run to their imc place in the
c     relevant array.
      izminmc(imc)=izmin
      izmaxmc(imc)=izmax

c     Calculate the average luminosity distance in each redshift bin.
         jmc=izmin,izmax
         dlarraymc(imc,jmc)=dlarray(jmc)/narray(jmc)


c     Set transfer the relevant variables for the run to their imc place in the
c     relevant array.
      izFRWmaxmc(imc)=izFRWmax

c     Calculate the average FRW redshift in each FRWredshift bin.
         jmc=1,izFRWmax
         redshiftFRWmc(imc,jmc)=redFRWarray(jmc)/nzFRWarray(jmc)


c     Set transfer the relevant variables for the run to their imc place in the
c     relevant array.
      ilammaxmc(imc)=ilammax

c     Calculate the average redshift and time tau in each lambda bin.
         jmc=1,ilammax
         redshiftmc(imc,jmc)=redarray(jmc)/nzarray(jmc)
         taumc(imc,jmc)=tauarray(jmc)/nzarray(jmc)


c     All the previous has been done for one imc run. This is repeated for
imc=1,nummc. Then ended, as below


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     8. Finally, we need to process all the data, for all the imc runs together. If
imc=1 this isn't relevant.
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     The following calculates the max izmax and min izmin from all the imc runs.
      izmaxmax=1
      izminmin=nbins
```

```
          imc=1,nummc
            (izmaxmax    izmaxmc(imc))izmaxmax=izmaxmc(imc) !maximum izmax
c           value reached , corresponding to maximum redshift reached in all imc
runs.
c           Start izmaxmax=1, so that can use condition if izmax(imc)<izmaxmax then
c           izmaxmax=izmax(imc), hence making izminmin the smallest.
            (izminmin    izminmc(imc))izminmin=izminmc(imc) !minimum izmin
c           value reached , corresponding to minimum redshift reached in all imc runs.
c           Start izminmin=nbins, so that can use condition if izminmin>izmin(imc) then
c           izminzmin=izmin(imc), hence making izminmin the smallest.

c        Now calculate:
c           dlav(jmc) = the sum of the average luminosity distance in each redshift
bin, over all the imc runs.
c           narraymc(jmc) = the number of imc runs that measured a redshift in that
bin .
c        Note that the average luminosity distance passed through per redshift bin,
over all the mc runs carried out is dlav(i)/narraymc(i).
             jmc=izminmc(imc),izmaxmc(imc)
             dlav(jmc)=dlav(jmc)+dlarraymc(imc,jmc)
             narraymc(jmc)=narraymc(jmc)+1



c     Similar interpretation to above, but for zFRW bins. Note that izFRWminmin=1 in
all cases.
      izFRWmaxmax=1
         imc=1,nummc
           (izFRWmaxmax    izFRWmaxmc(imc))izFRWmaxmax=
     &    izFRWmaxmc(imc)
           jmc=1,izFRWmaxmc(imc)
           redFRWdav(jmc)=redFRWdav(jmc)+redshiftFRWmc(imc,jmc)
           nzFRWarraymc(jmc)=nzFRWarraymc(jmc)+1



c     Similar interpretation to above, but for lambda bins
      ilammaxmax=1
         imc=1,nummc
           (ilammaxmax    ilammaxmc(imc))ilammaxmax=ilammaxmc(imc)
            jmc=1,ilammaxmc(imc)
            reddav(jmc)=reddav(jmc)+redshiftmc(imc,jmc)
            tauav(jmc)=tauav(jmc)+taumc(imc,jmc)
            nzarraymc(jmc)=nzarraymc(jmc)+1




c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     Now for some of the subroutines...
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     convert to spherical coordinates
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                cart2sphe(xi,yi,zi,ri,thetai,phii)

      real*8 xi,yi,zi,ri,thetai,phii

      ri=sqrt(xi**2d0+yi**2d0+zi**2d0)
        (xi    0d0      yi    0)
           (zi    0)
            thetai=0d0
            phii=0d0
```

```
               (zi    0.)
            thetai=dacos(-1d0)
            phii=0d0



      thetai=dacos(zi/ri)
        (yi    0)
           (xi    0d0)
            phii=0d0
               (xi    0d0)
            phii=dacos(-1d0)



         phii=dacos(xi/sqrt(ri**2d0-zi**2d0))
           (yi    0.)phii=2d0*dacos(-1d0)-phii





c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     invert the spherical polar transformation
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                 sphe2cart(r,theta,phi,x1,x2,x3)

      real*8 r,theta,phi,x1,x2,x3

      x1=r*dcos(phi)*dsin(theta)
      x2=r*dsin(phi)*dsin(theta)
      x3=r*dcos(theta)




c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     take dottau, dotr, dottheta, dotphi and use them to find nsold(i)
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                 vsphe2vnorm(r,theta,phi,dotr,dottheta,dotphi,
     &     dottau,nsold)

      real*8 r,theta,phi,dotr,dottau,
     &     dottheta,dotphi,nsold(3)
      real*8 A(3,3),det,sum
      real*8 E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,alphaR,alphaI,G,cc
             /params/ E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,
     & alphaR,alphaI,G,cc

      nsold(1)=dotr/dottau-cc*dsqrt(E-(1d0-2d0*G*m/r/(cc**2d0)))
      nsold(2)=dottheta/dottau
      nsold(3)=dotphi/dottau




c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     calculate nsnew(i) from the new spherical coordinates
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                 vnorm2vsphe(r,theta,phi,dotr,dottheta,dotphi,dottau,
     &     nsnew)

      real*8 r,theta,phi,dotr,dottau,
     &     dottheta,dotphi,nsnew(3)
      real*8 A(3,3),det,sum
      real*8 E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,alphaR,alphaI,G,cc
```

```
            /params/ E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,
     & alphaR,alphaI,G,cc

     dotr=dottau*(nsnew(1)+cc*dsqrt(E-(1d0-2d0*G*m/r/(cc**2d0))))
      dottheta=dottau*nsnew(2)
      dotphi=dottau*nsnew(3)




c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     calculates the n^a
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                vcart2vnorm(ri,thetai,phii,ncart,nsnew)

      real*8 ri,thetai,phii,dotxi,dotyi,dotzi,dotri,
     &     dotthetai,dotphii,ncart(3),nsnew(3)
      real*8 A(3,3),ai(3,3),det,sum

c     a is Jacobian for transformation
      a(1,1)=dcos(phii)*dsin(thetai)
      a(1,3)=-ri*dsin(phii)*dsin(thetai)
      a(1,2)=ri*dcos(phii)*dcos(thetai)
      a(2,1)=dsin(phii)*dsin(thetai)
      a(2,3)=ri*dcos(phii)*dsin(thetai)
      a(2,2)=ri*dsin(phii)*dcos(thetai)
      a(3,1)=dcos(thetai)
      a(3,3)=0d0
      a(3,2)=-ri*dsin(thetai)

c     det is the determinant of the Jacobian
      det=a(1,1)*a(2,2)*a(3,3)+a(1,2)*a(3,1)*a(2,3)
     & +a(1,3)*a(2,1)*a(3,2)
     &     -a(1,1)*a(3,2)*a(2,3)-a(1,2)*a(2,1)*a(3,3)
     &     -a(1,3)*a(3,1)*a(2,2)

c     ai is the inverse of a
      ai(1,1)=(a(2,2)*a(3,3)-a(3,2)*a(2,3))/det
      ai(2,1)=-(a(2,1)*a(3,3)-a(3,1)*a(2,3))/det
      ai(3,1)=(a(2,1)*a(3,2)-a(3,1)*a(2,2))/det
      ai(1,2)=-(a(1,2)*a(3,3)-a(3,2)*a(1,3))/det
      ai(2,2)=(a(1,1)*a(3,3)-a(3,1)*a(1,3))/det
      ai(3,2)=-(a(1,1)*a(3,2)-a(3,1)*a(1,2))/det
      ai(1,3)=(a(1,2)*a(2,3)-a(2,2)*a(1,3))/det
      ai(2,3)=-(a(1,1)*a(2,3)-a(2,1)*a(1,3))/det
      ai(3,3)=(a(1,1)*a(2,2)-a(2,1)*a(1,2))/det

c     defining dots in terms of n in cartesian.
      dotxi=ncart(1)
      dotyi=ncart(2)
      dotzi=ncart(3)

c     using equation (32) of CF
      dotri=ai(1,1)*dotxi+ai(1,2)*dotyi+ai(1,3)*dotzi
      dotthetai=ai(2,1)*dotxi+ai(2,2)*dotyi+ai(2,3)*dotzi
      dotphii=ai(3,1)*dotxi+ai(3,2)*dotyi+ai(3,3)*dotzi

c     calculate hat(n)^a in the new spherical coordinates.
      nsnew(1)=dotri
      nsnew(2)=dotthetai
      nsnew(3)=dotphii



c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
c      same as above, but does not return the normal vector
c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                  vcart2vsphe(ri,thetai,phii,dotxi,dotyi,dotzi,
     &     dotri,dotthetai,dotphii)

       real*8 ri,thetai,phii,dotxi,dotyi,dotzi,dotri,
     &     dotthetai,dotphii
       real*8 A(3,3),ai(3,3),det,sum

c      a is Jacobian for transformation
       a(1,1)=dcos(phii)*dsin(thetai)
       a(1,3)=-ri*dsin(phii)*dsin(thetai)
       a(1,2)=ri*dcos(phii)*dcos(thetai)
       a(2,1)=dsin(phii)*dsin(thetai)
       a(2,3)=ri*dcos(phii)*dsin(thetai)
       a(2,2)=ri*dsin(phii)*dcos(thetai)
       a(3,1)=dcos(thetai)
       a(3,3)=0d0
       a(3,2)=-ri*dsin(thetai)

c      det is the determinant of the Jacobian
       det=a(1,1)*a(2,2)*a(3,3)+a(1,2)*a(3,1)*a(2,3)
     & +a(1,3)*a(2,1)*a(3,2)
     &     -a(1,1)*a(3,2)*a(2,3)-a(1,2)*a(2,1)*a(3,3)
     &     -a(1,3)*a(3,1)*a(2,2)

c      ai is the inverse of a
       ai(1,1)=(a(2,2)*a(3,3)-a(3,2)*a(2,3))/det
       ai(2,1)=-(a(2,1)*a(3,3)-a(3,1)*a(2,3))/det
       ai(3,1)=(a(2,1)*a(3,2)-a(3,1)*a(2,2))/det
       ai(1,2)=-(a(1,2)*a(3,3)-a(3,2)*a(1,3))/det
       ai(2,2)=(a(1,1)*a(3,3)-a(3,1)*a(1,3))/det
       ai(3,2)=-(a(1,1)*a(3,2)-a(3,1)*a(1,2))/det
       ai(1,3)=(a(1,2)*a(2,3)-a(2,2)*a(1,3))/det
       ai(2,3)=-(a(1,1)*a(2,3)-a(2,1)*a(1,3))/det
       ai(3,3)=(a(1,1)*a(2,2)-a(2,1)*a(1,2))/det

       dotri=ai(1,1)*dotxi+ai(1,2)*dotyi+ai(1,3)*dotzi
       dotthetai=ai(2,1)*dotxi+ai(2,2)*dotyi+ai(2,3)*dotzi
       dotphii=ai(3,1)*dotxi+ai(3,2)*dotyi+ai(3,3)*dotzi




c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c      finds ncart
c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                  vnorm2vcart(ri,thetai,phii,nsold,ncart)

       real*8 ri,thetai,phii,dotxi,dotyi,dotzi,dotri,
     &     dotthetai,dotphii,nsold(3),ncart(3)
       real*8 A(3,3),det,sum
       integer i,j,k

c      a is Jacobian for transformation
       a(1,1)=dcos(phii)*dsin(thetai)
       a(1,3)=-ri*dsin(phii)*dsin(thetai)
       a(1,2)=ri*dcos(phii)*dcos(thetai)
       a(2,1)=dsin(phii)*dsin(thetai)
       a(2,3)=ri*dcos(phii)*dsin(thetai)
       a(2,2)=ri*dsin(phii)*dcos(thetai)
       a(3,1)=dcos(thetai)
       a(3,3)=0d0
       a(3,2)=-ri*dsin(thetai)
```

```
      dotri=nsold(1)
      dotthetai=nsold(2)
      dotphii=nsold(3)

      dotxi=a(1,1)*dotri+a(1,2)*dotthetai+a(1,3)*dotphii
      dotyi=a(2,1)*dotri+a(2,2)*dotthetai+a(2,3)*dotphii
      dotzi=a(3,1)*dotri+a(3,2)*dotthetai+a(3,3)*dotphii

      ncart(1)=dotxi
      ncart(2)=dotyi
      ncart(3)=dotzi




c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     find cartesian velocity from spherical
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                 vsphe2vcart(ri,thetai,phii,
     &     dotri,dotthetai,dotphii,dotxi,dotyi,dotzi)

      real*8 ri,thetai,phii,dotxi,dotyi,dotzi,dotri,
     &     dotthetai,dotphii
      real*8 A(3,3),det,sum
      integer i,j,k

c     a is Jacobian for transformation
      a(1,1)=dcos(phii)*dsin(thetai)
      a(1,3)=-ri*dsin(phii)*dsin(thetai)
      a(1,2)=ri*dcos(phii)*dcos(thetai)
      a(2,1)=dsin(phii)*dsin(thetai)
      a(2,3)=ri*dcos(phii)*dsin(thetai)
      a(2,2)=ri*dsin(phii)*dcos(thetai)
      a(3,1)=dcos(thetai)
      a(3,3)=0d0
      a(3,2)=-ri*dsin(thetai)

      dotxi=a(1,1)*dotri+a(1,2)*dotthetai+a(1,3)*dotphii
      dotyi=a(2,1)*dotri+a(2,2)*dotthetai+a(2,3)*dotphii
      dotzi=a(3,1)*dotri+a(3,2)*dotthetai+a(3,3)*dotphii




c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     calculate the first integrals
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                 integrals(ri,thetai,phii,dotri,dotthetai,dotphii,
     &     dottaui)

      real*8 ri,thetai,phii,dotri,dotthetai,dotphii,dottaui
      real*8 E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,alphaR,alphaI,G,cc
             /params/ E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,
     & alphaR,alphaI,G,cc

      Jphi=m*ri**2d0*(dsin(thetai))**2d0*dotphii

      J=m*ri**2d0*
     & dsqrt(dotthetai**2d0+dsin(thetai)**2d0*dotphii**2d0)

      B=dsqrt(1d0/E*((m*cc*dotri)**2d0+
     & (J*cc/ri**2d0)**2d0*(1d0-2d0*G*m/ri/(cc**2d0))))

      dottaui=(E*B-m*dotri*dsqrt(E-(1d0-2d0*G*m/ri/(cc**2d0))))
```

```
     & /(m*cc**2d0*(1d0-2d0*G*m/ri/(cc**2d0)))



c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c      calculates the t vector
c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                  createtb(xb,ri,thetai,phii,dotri,dotthetai,dotphii,
     &     dottaui,tcartb)

      integer i
      real*8 ri,thetai,phii,dotri,dotthetai,dotphii,dottaui,pi,
     &     tcartb(4,2),ttemp(4,2),nb(3),xb,k(4),c(4),d(4),a(3,3)
      real*8 E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,alphaR,alphaI,G,cc
             /params/ E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,
     & alphaR,alphaI,G,cc

c      calculate the nb.
      nb(1)=dotri/dottaui-cc*dsqrt(E-(1d0-2d0*G*m/ri/(cc**2d0)))
      nb(2)=dotthetai/dottaui
      nb(3)=dotphii/dottaui

c      defines k, the vector tangent to null geodesic
      k(1)=dottaui
      k(2)=dottaui*(cc*dsqrt(E-(1d0-2d0*G*m/ri/(cc**2d0))+nb(1)))
      k(3)=dottaui*nb(2)
      k(4)=dottaui*nb(3)

c      the vector c is the real part of tb
      c(1)=0d0
      c(2)=0d0
      c(3)=-ri*dottaui*dsin(thetai)*nb(3)/dsqrt(2d0)/J
      c(4)=ri*dottaui*nb(2)/dsqrt(2d0)/J/dsin(thetai)

c      the vector d is the imaginary part of tb (see eq 76)
      d(1)=ri*dottaui*nb(1)/dsqrt(2d0)/J/dsqrt(E)
      d(2)=ri*dottaui*(dsqrt(E)+dsqrt(E-(1d0-2d0*G*m/ri/(cc**2d0))
     & *nb(1)/dsqrt(E))/dsqrt(2d0)/J)
      d(3)=0d0
      d(4)=0d0

c      calculate alpha at end of propagation (eq 79). Note alphaR=constant.
      alphaI=alphaI-B*(xb-lambdai)/dsqrt(2d0)/J

c      set geodesic parameter lambda to the final time tau after propagating:
      lambdai=xb

c      the following calculates the real part (ttemp(i,1)) and complex part
c      (ttemp(i,2)) of the tangent vector t. Note ppsi0=psi/2 in this case
         i=1,4
         ttemp(i,1)=dcos(ppsi0)*(c(i)+alphaR*k(i))
     &        +dsin(ppsi0)*(d(i)+alphaI*k(i))
         ttemp(i,2)=-dsin(ppsi0)*(c(i)+alphaR*k(i))
     &        +dcos(ppsi0)*(d(i)+alphaI*k(i))


      a(1,1)=dcos(phii)*dsin(thetai)
      a(1,3)=-ri*dsin(phii)*dsin(thetai)
      a(1,2)=ri*dcos(phii)*dcos(thetai)
      a(2,1)=dsin(phii)*dsin(thetai)
      a(2,3)=ri*dcos(phii)*dsin(thetai)
      a(2,2)=ri*dsin(phii)*dcos(thetai)
      a(3,1)=dcos(thetai)
      a(3,3)=0d0
      a(3,2)=-ri*dsin(thetai)
```

```
c     converts ttemp into cartesian coordinates. Note first components are
c     time, so do not get transformed.
      tcartb(1,1)=ttemp(1,1)
      tcartb(2,1)=a(1,1)*ttemp(2,1)+a(1,2)*ttemp(3,1)
     & +a(1,3)*ttemp(4,1)
      tcartb(3,1)=a(2,1)*ttemp(2,1)+a(2,2)*ttemp(3,1)
     & +a(2,3)*ttemp(4,1)
      tcartb(4,1)=a(3,1)*ttemp(2,1)+a(3,2)*ttemp(3,1)
     & +a(3,3)*ttemp(4,1)
      tcartb(1,2)=ttemp(1,2)
      tcartb(2,2)=a(1,1)*ttemp(2,2)+a(1,2)*ttemp(3,2)
     & +a(1,3)*ttemp(4,2)
      tcartb(3,2)=a(2,1)*ttemp(2,2)+a(2,2)*ttemp(3,2)
     & +a(2,3)*ttemp(4,2)
      tcartb(4,2)=a(3,1)*ttemp(2,2)+a(3,2)*ttemp(3,2)
     & +a(3,3)*ttemp(4,2)



c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     calculate the optical properties like shear etc
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                calcalphappsi(xb,ri,thetai,phii,dotri,dotthetai,
     &     dotphii,dottaui,tcarta)

      real*8 ri,thetai,phii,dotri,dotthetai,dotphii,dottaui,
     &     ttemp(4,2),tcarta(4,2),nb(3),xb,k(4),p(2),q(2),d(2)
      real*8 aa,bb,cR,cI,det,num,num2,den,a(3,3),ai(3,3),pi,dd
      real*8 E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,alphaR,alphaI,G,cc
        /params/ E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,
     & alphaR,alphaI,G,cc

      a(1,1)=dcos(phii)*dsin(thetai)
      a(1,3)=-ri*dsin(phii)*dsin(thetai)
      a(1,2)=ri*dcos(phii)*dcos(thetai)
      a(2,1)=dsin(phii)*dsin(thetai)
      a(2,3)=ri*dcos(phii)*dsin(thetai)
      a(2,2)=ri*dsin(phii)*dcos(thetai)
      a(3,1)=dcos(thetai)
      a(3,3)=0d0
      a(3,2)=-ri*dsin(thetai)

      det=a(1,1)*a(2,2)*a(3,3)+a(1,2)*a(3,1)*a(2,3)
     &     +a(1,3)*a(2,1)*a(3,2)
     &     -a(1,1)*a(3,2)*a(2,3)-a(1,2)*a(2,1)*a(3,3)
     &     -a(1,3)*a(3,1)*a(2,2)

      ai(1,1)=(a(2,2)*a(3,3)-a(3,2)*a(2,3))/det
      ai(2,1)=-(a(2,1)*a(3,3)-a(3,1)*a(2,3))/det
      ai(3,1)=(a(2,1)*a(3,2)-a(3,1)*a(2,2))/det
      ai(1,2)=-(a(1,2)*a(3,3)-a(3,2)*a(1,3))/det
      ai(2,2)=(a(1,1)*a(3,3)-a(3,1)*a(1,3))/det
      ai(3,2)=-(a(1,1)*a(3,2)-a(3,1)*a(1,2))/det
      ai(1,3)=(a(1,2)*a(2,3)-a(2,2)*a(1,3))/det
      ai(2,3)=-(a(1,1)*a(2,3)-a(2,1)*a(1,3))/det
      ai(3,3)=(a(1,1)*a(2,2)-a(2,1)*a(1,2))/det

c     convert ttemp to spherical coordinates
      ttemp(1,1)=tcarta(1,1)
      ttemp(2,1)=ai(1,1)*tcarta(2,1)+ai(1,2)*tcarta(3,1)
     &     +ai(1,3)*tcarta(4,1)
      ttemp(3,1)=ai(2,1)*tcarta(2,1)+ai(2,2)*tcarta(3,1)
     &     +ai(2,3)*tcarta(4,1)
      ttemp(4,1)=ai(3,1)*tcarta(2,1)+ai(3,2)*tcarta(3,1)
     &     +ai(3,3)*tcarta(4,1)
      ttemp(1,2)=tcarta(1,2)
```

```
1060        ttemp(2,2)=ai(1,1)*tcarta(2,2)+ai(1,2)*tcarta(3,2)
           &     +ai(1,3)*tcarta(4,2)
            ttemp(3,2)=ai(2,1)*tcarta(2,2)+ai(2,2)*tcarta(3,2)
           &     +ai(2,3)*tcarta(4,2)
            ttemp(4,2)=ai(3,1)*tcarta(2,2)+ai(3,2)*tcarta(3,2)
           &     +ai(3,3)*tcarta(4,2)

      c     as before
            nb(1)=dotri/dottaui-cc*dsqrt(E-(1d0-2d0*G*m/ri/(cc**2d0)))
            nb(2)=dotthetai/dottaui
1070        nb(3)=dotphii/dottaui

      c     same k tangent vector
            k(1)=dottaui
            k(2)=dottaui*(cc*dsqrt(E-(1d0-2d0*G*m/ri/(cc**2d0)))+nb(1))

      c     p is real part of t vector, and q the imaginary part.
            p(1)=ttemp(1,1)
            p(2)=ttemp(2,1)
            q(1)=ttemp(1,2)
1080        q(2)=ttemp(2,2)
            d(1)=ri*nb(1)/dsqrt(2d0)/J !Error: NEED DOTTAU
            d(2)=ri*(1d0+dsqrt(2d0*m/ri)*nb(1))/dsqrt(2d0)/J !Error: NEED DOTTAU

            aa=p(2)*k(1)-p(1)*k(2)
            bb=q(2)*k(1)-q(1)*k(2)
            cR=q(2)*d(1)-q(1)*d(2)
            cI=p(1)*d(2)-p(2)*d(1)
            det=aa**2d0+bb**2d0

1090        alphaR=(aa*cR+bb*cI)/det
            alphaI=(-bb*cR+aa*cI)/det

            num=q(1)*alphaR*k(1)-p(1)*(d(1)+alphaI*k(1))
            den=p(1)*alphaR*k(1)+q(1)*(d(1)+alphaI*k(1))

            dd=alphaR**2d0*(k(1))**2d0+(d(1))**2d0
           & +2d0*(d(1))*alphaI*(k(1))+alphaI**2d0*(k(1))**2d0

            det=dsqrt(num**2d0+den**2d0)
1100        ppsi0=dacos(den/det)

              (num    0.)
                    (dd    0.)
                        ppsi0=ppsi0-dacos(-1d0)
                        (dd    0.)
                        ppsi0=ppsi0

                   (num    0.)
                     (dd    0.)
1110                    ppsi0=dacos(-1d0)-ppsi0
                        (dd    0.)
                        ppsi0=-ppsi0


            ppsi0=2d0*ppsi0




1120  c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      c     integrates equations using dverk. xa=independent variable.
      c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                        integrator(n_eq,derivs,xa,y,dt,tol,ind,c,nw,w)

            real*8 tol,tol1
```

```
      integer n_eq,nw,ind,i
      real*8 c(nw+30),w(nw,18)
      real*8 y(n_eq)
1130  real*8 dt,xa,xb,xi,xf
      real*8 E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,alphaR,alphaI,G,cc
     &       /params/ E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,
     & alphaR,alphaI,G,cc

      derivs

      xi=xa
      xf=xa+dt
      tol1=tol
1140  ind = 1

  500 (ind    -1)
      tol1=tol1*2d0
      ind = 3


       dverk(n_eq,derivs,xi,y,xf,tol1,ind,c,nw,w)
      (ind    4)
           (6,*)"about to return"
1150
           dverk(n_eq,derivs,xi,y,xf,tol1,ind,c,nw,w)


      (ind    -1)      500



c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     calculates the equations to use in the integrator
1160  c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            derivs(n,xx,y,y_dot,identity)

      integer n, identity
      real*8 y(n),y_dot(n),xx
      real*8 r,theta,phi,tau,dotr,dottheta,dotphi,dottau,dotpphi,
     &     dotdotr,dotdottheta,dotra,sigma,ra,dotsigma,dotdotra,
     &     pphi
      real*8 E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,alphaR,alphaI,G,cc
     &       /params/ E,m,B,J,Jphi,tau0,ppsi0,lambdai,sigma0,
1170  & alphaR,alphaI,G,cc

c     sets r, theta, ... to values obtained from last integration
      r=y(1)
      theta=y(2)
      phi=y(3)
      tau=y(4)
      dotr=y(5)
      dottheta=y(6)
      sigma=y(7)
1180  ra=y(8)

      dotphi=Jphi/r**2d0/(dsin(theta))**2d0/m

      dottau=(E*B-m*dotr*cc*dsqrt(E-(1d0-2d0*G*m/r/(cc**2d0))))
     & /((1d0-2d0*G*m/r/(cc**2d0))*m*cc**2d0)

      dotdotr=(r-3d0*G*m/cc**2d0)*
     & (dottheta**2d0+dsin(theta)**2d0*dotphi**2d0)

1190  dotdottheta=dsin(theta)*dcos(theta)*dotphi**2d0
     &-2d0*dottheta*dotr/r

      dotra=y(9)
```

```
      pphi=y(10)

      dotsigma=3d0*m*J**2d0*ra**2d0/r**5d0*dcos(ppsi0-pphi) !(69)
      dotdotra=-sigma**2d0/ra**3d0 !(65)
      dotpphi=3d0*m*J**2d0*ra**2d0/r**5d0*dsin(ppsi0-pphi)/sigma !(70)

c     define the initial y_dot values to go into dverk
      y_dot(1)=dotr
      y_dot(2)=dottheta
      y_dot(3)=dotphi
      y_dot(4)=dottau
      y_dot(5)=dotdotr
      y_dot(6)=dotdottheta
      y_dot(7)=dotsigma
      y_dot(8)=dotra
      y_dot(9)=dotdotra
      y_dot(10)=dotpphi




c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     The following subroutines calculate the hyperbolic volume h_vol. This code
c     is adapted from Numerical Recipes. See this book for more details
c     about the code
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                hvolcalc(vol)

      real*8 s,func,x1,x2,pi,vol
             func
      pi=acos(-1d0)
      x1=-pi/4d0
      x2=pi/4d0
           quad3d(x1,x2,s)
      vol=6d0*s
           (6,*)vol


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                quad3d(x1,x2,ss)
      REAL*8 ss,x1,x2,h
             h
c      USES h,qgausx
c     Returns as ss the integral of a user-supplied function func over a three-
dimensional region
c     specified by the limits x1, x2, and by the user-supplied functions y1, y2, z1,
and z2, as
c     defined in (4.6.2).
           qsimpx(h,x1,x2,ss)
c      write(6,*)"ss",ss


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

             f(zz)
      REAL*8 f,zz,func,x,y,z
            /xyz/ x,y,z
c     USES func
c     Called by qgausz. Calls func.
      z=zz
      f=func(x,y,z)
c      write(6,*)"f",f


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
               g(yy)
      REAL*8 g,yy,f,z1,z2,x,y,z
               f
             /xyz/ x,y,z
c     USES f,qgausz,z1,z2
c     Called by qgausy. Calls qgausz.
      REAL*8 ss
      y=yy
           qsimpz(f,z1(x,y),z2(x,y),ss)
      g=ss
c      write(6,*)"g",g


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

               h(xx)
      REAL*8 h,xx,g,y1,y2,x,y,z
               g
             /xyz/ x,y,z
c     USES g,qgausy,y1,y2
c     Called by qgausx. Calls qgausy.
      REAL*8 ss
      x=xx
           qsimpy(g,y1(x),y2(x),ss)
      h=ss



c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

               func(phi,theta,r)
      REAL*8  r,theta,phi,func
      REAL*8  sin
                sin
      func = r**2d0*sin(theta)/(1d0-r**2d0)**3d0


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

               y1(phi)
      REAL*8  phi,y1,pi
      pi=acos(-1d0)
      y1=pi/4d0


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

               y2(phi)
      REAL*8  phi,y2,pi
      pi=acos(-1d0)
      y2=3d0*pi/4d0


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

               z1(phi,theta)
      REAL*8  theta,phi,z1
      z1 = 0d0


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

               z2(phi,theta)
      REAL*8  r,theta,phi,z2
      z2 = sin(theta)*cos(phi)*sqrt(sqrt(5d0)+2d0)
     & -sqrt(-1d0+(2d0+sqrt(5d0))*((sin(theta))**2d0)
```

```
     & *(cos(phi))**2d0)


c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                 qsimpx(funct,a,b,s)
      INTEGER JMAX
      REAL*8 a,b,funct,s,EPS
               funct
      PARAMETER (EPS=1d-6,JMAX=20)
c      USES trapzd
c      Returns as s the integral of the dvolfunction dvolfunc from a to b. The
c     parameters EPS can be set to the desired fractional accuracy and JMAX so
c     that 2 to the power JMAX-1 is the maximum allowed number of steps.
c     Integration is performed by Simpson’s rule.
      INTEGER j
      REAL*8 os,ost,st
      ost=1d-30
      os= 1d-30
         j=1,JMAX
             trapzd(funct,a,b,st,j)
        s=(4d0*st-ost)/3d0
c      Compare equation (4.2d04), above.
           (j    5)
c     Avoid spurious early convergence.
             (abs(s-os)    EPS*abs(os)    (s    0d0     os    0.))
     &

        os=s
        ost=st

c      pause 'too many steps in qsimp'

c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                 qsimpy(funct,a,b,s)
      INTEGER JMAX
      REAL*8 a,b,funct,s,EPS
               funct
      PARAMETER (EPS=1d-6,JMAX=20)
c      USES trapzd
c      Returns as s the integral of the dvolfunction dvolfunc from a to b. The
c     parameters EPS can be set to the desired fractional accuracy and JMAX so
c     that 2 to the power JMAX-1 is the maximum allowed number of steps.
c     Integration is performed by Simpson’s rule.
      INTEGER j
      REAL*8 os,ost,st
      ost=1d-30
      os= 1d-30
         j=1,JMAX
             trapzd(funct,a,b,st,j)
        s=(4d0*st-ost)/3d0
c      Compare equation (4.2d04), above.
           (j    5)
c     Avoid spurious early convergence.
             (abs(s-os)    EPS*abs(os)    (s    0d0     os    0.))
     &

        os=s
        ost=st

c      pause 'too many steps in qsimp'

c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                 qsimpz(funct,a,b,s)
      INTEGER JMAX
```

```
      REAL*8 a,b,funct,s,EPS
             funct
      PARAMETER (EPS=1d-6,JMAX=20)
c      USES trapzd
c      Returns as s the integral of the dvolfunction dvolfunc from a to b. The
c     parameters EPS can be set to the desired fractional accuracy and JMAX so
c     that 2 to the power JMAX-1 is the maximum allowed number of steps.
c     Integration is performed by Simpson's rule.
      INTEGER j
      REAL*8 os,ost,st
      ost=1d-30
      os= 1d-30
         j=1,JMAX
            trapzd(funct,a,b,st,j)
        s=(4d0*st-ost)/3d0
c      Compare equation (4.2d04), above.
          (j    5)
c     Avoid spurious early convergence.
            (abs(s-os)    EPS*abs(os)    (s    0d0     os    0.))
     &

        os=s
        ost=st

c      pause 'too many steps in qsimp'

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                  trapzd(funct,a,b,s,n)
      INTEGER n
      REAL*8 a,b,s,funct
             funct
c      This routine computes the nth stage of refinement of an extended
c     trapezoidal rule. dvolfunc is input as the name of the dvolfunction to be
c     integrated between limits a and b, also input. When b called with n=1,
c     the routine returns as s the crudest estimate of a f (x)dx. Subsequent
c     calls with n=2,3,... (in that sequential order) will improve the
c     accuracy of s by adding 2n-2 additional interior points. s should not be
c     modified between sequential calls.
      INTEGER it,j
      REAL*8 del,summ,tnm,x,val
         (n    1)
        s=5d-1*(b-a)*(funct(a)+funct(b))

        it=2d0**(n-2d0)
        tnm=it
        del=(b-a)/tnm
c      This is the spacing of the points to be added.
        x=a+5d-1*del
        summ=0d0
           j=1,it
          summ=summ+funct(x)
          x=x+del

        s=5d-1*(s+(b-a)*summ/tnm)
      val=funct(b)
c      This replaces s by its refined value.


c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     This subroutine will find the value of a and eta corresponding to a given time
     tau
c     given the value of K and a0. eta_min and eta_max are the bounds on the
c     value of eta that is used for the numerical solution. When using in a program,
c     determine eta_min and eta_max from last value of eta so is quick to solve.
```

```
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1460
                  find_a_from_tau(const1,cc,tau,eta_min,eta_max,a,eta)

      REAL*8 tau,eta_min,eta_max,a,const1,pi,a_const
      real*8 eta_acc,eta_relacc,eta,eta_est,tau_func,cc
               tau_func,eta_est

c      eta_acc is the accuracy in eta desired, and eta_relacc is the relative
c     accuracy in eta desired.
      eta_acc=1d-12
1470  eta_relacc=1d-12

c     use the function eta_est (the numerical solver) to find eta
      eta=eta_est(const1,cc,tau,tau_func,eta_min,eta_max,eta_acc,
     & eta_relacc)
c      write(6,*)"eta",eta
c     solve for a from eta
      a=const1*(dcosh(eta)-1d0)
           (6,*)"eta",eta,"tau",tau

1480



c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     eta_est is a function returning the numerical estimate for eta as a solution
c     to the equation of tau(eta)=tau, where tau is a known value.
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                  eta_est(const1,cc,tau,tau_func,eta_min,eta_max,
     & eta_acc,eta_relacc)
1490  INTEGER j,JMAX
      REAL*8 eta_est,eta_min,eta_max,eta_acc,eta_relacc,deta,f,df
      real*8 const1,tau,cc
      PARAMETER (JMAX=20)
      eta_est=(5d-1)*(eta_min+eta_max)
         j=1,JMAX
c     call the subroutine tau_func, which returns the value of tau for a given
c     eta (in this case eta_est, the estimated value)
           tau_func(const1,cc,tau,eta_est,f,df)

1500  c     the newton raphson method:
        deta=f/df
        eta_est=eta_est-deta
          ((eta_min-eta_est)*(eta_est-eta_max)    0.)
             (6,*) "eta_est = ",eta_est," jumped out of brackets"

        (abs(deta)    eta_acc)


        (abs(deta)/eta_est    eta_relacc)
1510



             (6,*) "rtnewt exceeded maximum iterations"



c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     subroutine for eta_est
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1520                tau_func(const1,cc,tau,eta,fn,df)

      REAL*8 eta,fn,df,tau,const1,cc

c     calculate value of eta such that tau(eta)=tau, or tau(eta)-tau=0
      fn=const1/cc*(dsinh(eta)-eta)-tau
```

```
c      calculate derivative of tau(eta)
       df=const1/cc*(dcosh(eta)-1d0)




c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c      A random number generator
c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                ran1(idum)
       INTEGER idum,IA,IM,IQ,IR,NTAB,NDIV
       REAL ran1,AM,EPS,RNMX
       PARAMETER (IA=16807,IM=2147483647,AM=1d0/IM,IQ=127773,IR=2836,
      *NTAB=32,NDIV=1+(IM-1)/NTAB,EPS=1.2e-7,RNMX=1d0-EPS)
       INTEGER j,k,iv(NTAB),iy
            iv,iy
            iv /NTAB*0/, iy /0/
          (idum    0    iy    0)
         idum=max(-idum,1)
            11 j=NTAB+8,1,-1
           k=idum/IQ
           idum=IA*(idum-k*IQ)-IR*k
              (idum    0) idum=idum+IM
              (j    NTAB) iv(j)=idum
11
         iy=iv(1)

       k=idum/IQ
       idum=IA*(idum-k*IQ)-IR*k
          (idum    0) idum=idum+IM
       j=1+iy/NDIV
       iy=iv(j)
       iv(j)=idum
       ran1=min(AM*iy,RNMX)



c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

              'dverk.f'

c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c      The end
c      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```